VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

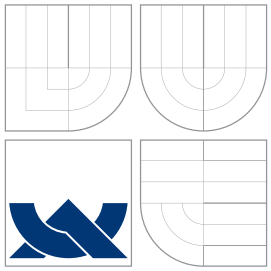
INFORMAČNÍ SYSTÉM PRO PODPORU PROJEKTOVÉHO ŘÍZENÍ A TÝMOVOU SPOLUPRÁCI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

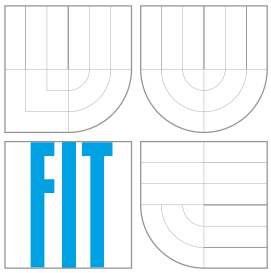
AUTOR PRÁCE
AUTHOR

TOMÁŠ VÍTEK

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO PODPORU PROJEKTOVÉHO ŘÍZENÍ A TÝMOVOU SPOLUPRÁCI

INFORMATION SYSTEM FOR PROJECT MANAGEMENT AND COLLABORATION SUPPORT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — TOMÁŠ VÍTEK
AUTHOR

VEDOUCÍ PRÁCE — Mgr. MAREK RYCHLÝ
SUPERVISOR

BRNO 2007

Abstrakt

Projektové řízení je pojem, který stále více společností zařazuje do svého slovníku. Ostrý konkurenční boj nutí firmy efektivněji pracovat a přistupovat k individuálním zakázkám jako samostatným projektům. Na těchto projektech běžně spolupracují lidé z různých oddělení firmy, projektu se přímo účastní zástupci subdodavatelů i odběratele produktu. Prolínají se tak dříve striktně oddělené oblasti softwarových produktů – nástrojů projektového řízení a on-line aplikací podporujících týmovou spolupráci.

Rozvoj širokopásmového připojení k internetu změnil v mnoha organizacích způsob práce i využívání informačních technologií. Díky všudypřítomnému internetovému připojení se prosazuje pronájem software metodou Software as a Service (SaaS).

Cílem této práce je navrhnout informační systém, který podpoří projektové řízení ve vybrané organizaci a umožní efektivní komunikaci členů projektového týmu.

Klíčová slova

projektové řízení, týmová spolupráce, informační systémy, Java Enterprise Edition, Spring Framework, objektově relační mapování

Abstract

Project management is used in increasing number of companies. Competition on the market requires companies to work more efficiently and approach separate orders as independent projects. Teams that work on the projects consist of employees from different departments, from subcontractors and from the client as well.

Rapid development of broadband internet connection changed work methods and use of information technology. Thanks to ubiquitous internet connection software application delivery model known as Software as a Service (SaaS) is gaining on popularity.

Main aim of this project is to design an information system that would support project management and allow effective team communication and collaboration.

Keywords

project management, collaboration software, infromation systems, Java Enterprise Edition, Spring Framework, object-relational mapping

Citace

Tomáš Vítek: Informační systém pro podporu projektového řízení a týmovou spolupráci, diplomová práce, Brno, FIT VUT v Brně, 2007

Informační systém pro podporu projektového řízení a týmovou spolupráci

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Marka Rychlého.

.....
Tomáš Vítek
22. května 2007

Poděkování

Rád bych zde poděkoval všem, kteří měli vliv na vznik této práce. Jmenovitě pak Markovi Rychlému za odborné vedení a poskytnuté rady, Bentu Hansenovi za zajímavé přednášky o projektovém řízeni na University of Southern Denmark, které výrazně rozšířily mé znalosti a ovlivnily výsledný návrh systému. V neposlední řadě Pavlovi Hudranovi za pomoc při zpracování CSS stylů implementovaného systému.

Obsah

Předmluva

Po téměř celou dobu mého studia na vysoké škole jsem byl členem brněnského centra neziskové studentské organizace IAESTE České republiky. V roce 2002 jsem začal příležitostně pomáhat na dílčích úkolech a postupně pracoval na projektech s větší odpovědností. Své aktivní členství jsem končil jako vedoucí lokálního centra a projektový manažer nejdůležitějšího komerčního projektu IAESTE ČR – Katalogu pracovních příležitostí. V průběhu těch několika let jsem dobře poznal práci v této organizaci, ať už z pozice řadového člena, tak z pozice koordinátora projektu, který je zodpovědný za vedení týmu studentů.

Několik roků jsme vybírali a zkoušeli používat vhodný informační systém, který by v naší, v mnoha ohledech netypické, organizaci prospěl při vedení projektů i vzájemné komunikaci členů. Z obrovského množství existujících systémů, které se lišily rozsahem funkcí, filozofií ovládání nebo cenou, se pro potřeby naší organizace nehodil žádný. Proto jsem se rozhodl jako téma mé závěrečné diplomové práce na Fakultě informačních technologií navrhnout a implementovat systém, který usnadní práci nejen v IAESTE České republiky, ale i v dalších obdobných organizacích.

Kapitola 1

Úvod

Tato práce se zabývá problematikou softwarové podpory projektového řízení a týmové spolupráce. Protože se jedná o často skloňované a zároveň obecné termíny, je nezbytné definovat jejich význam a praktické uplatnění. První kapitola popisuje oba tyto termíny a možnosti jejich podpory prostřednictvím informačních technologií. Představuje rovněž studentskou organizaci IAESTE České republiky, která slouží jako vzorová organizace, ve které je vhodné nasadit informační systém podporující řízení projektů a efektivní týmovou spolupráci.

Druhá a třetí kapitola specifikuje a analyzuje požadavky na informační systém, na dílčí nástroje a zajištění jejich vzajemné provázanosti. Pátá kapitola popisuje navržený systém s pomocí UML notace.

Část nazvaná Implementace se zabývá výběrem vhodné implementační technologie a podpůrných nástrojů, jako jsou webové frameworky nebo nástroje zajišťující objektově relační mapování. Popisuje také implementaci nejdůležitějších částí výsledného informačního systému.

Polední kapitola se věnuje dalším možnostem rozšíření informačního systému.

1.1 Projektové řízení

Podle [4] je *projekt* organizovaná dočasná spolupráce mezi lidmi zaměřená na vytvoření jedinečného produktu nebo služby za určených podmínek a s omezenými zdroji. Tato věta vyjadřuje hlavní podstatu projektu a projektového řízení. Každý projekt je časově omezený (má určený začátek a konec), jedinečný (nejedná se o opakovanou činnost, výrobu nebo poskytování stejné služby) a má omezené zdroje (finance, lidské zdroje, čas, materiály . . .).

Projektové řízení je oblast managementu zaměřená na organizaci a řízení zdrojů takovým způsobem, aby bylo dosaženo včasného dokončení projektu v celém svém rozsahu a při dodržení definovaných omezení. [4]

Základní omezení, se kterými se v projektovém řízení setkáváme se nazývají podle Rosenau [7] The Triple Constraint[1]. The Triple Constraint definuje tři omezení, která se navzájem ovlivňují a změna jednoho omezení má vliv na ostatní omezení. Jedná se o čas, zdroje a rozsah projektu. Pro názornost bývají zobrazeny jako trojúhelník, jehož strany tvoří jednotlivá omezení. Projektový management poskytuje nástroje a metody, které umožní projektovému týmu (ne pouze projektovému manažerovi) zorganizovat práci v rámci daných omezení.

[1] V českém překladu [8] je použit termín *trojimperativ*.

Protože za projekt považujeme nejrůznější činnosti, které splňují výše uvedenou definici, je oblast projektového řízení velmi široká. Za projekt můžeme považovat zorganizování večírku, vývoj nového produktu nebo celý projekt Apollo. Práce projektového manažera se proto může velmi lišit v nezbytných úkonech a oblastech zodpovědnosti. Obecně existuje pět základních oblastí, ze kterých se skládá řízení každého projektu. Jedná se o definování, plánování, vedení, kontrolování a uzavření projektu.

Definice cílů projektu je první oblastí projektového řízení. Cílem tohoto kroku je přesně definovat, analyzovat a vymezit projekt. Tato často podceňovaná část má značný vliv na celý proces zpracování projektu. Definice cílů projektu zahrnuje oficiální zadání projektu, včetně definovaných omezení (The Tripple Constraint), dále analýzu aktuální situace a historie projektu, vymezení všech zainteresovaných stran, jejich očekávání a vlivu na projekt. Vhodným nástrojem analýzy je tzv. 5×5 model[4], poskytující seznam oblastí, které je nutné do analýzy zahrnout.

Výsledkem tohoto kroku by mělo být plné porozumění zadání projektu, očekávání od projektu a rovněž všech okolností, které mají na projekt přímý nebo nepřímý vliv. Dobře analyzované zadání projektu je základem pro další krok projektového řízení – plánování projektu.

Plánování projektu a jeho smysl nejlépe vystihuje Milton D. Rosenau[7] v úvodu ke kapitole o plánování projektu:

> Plánovací činnosti jsou pro řízení projektu rozhodující. Plány jsou simulací projektu, protože obsahují písemný popis toho, jak budou splněny parametry „trojimperativu“. Proto jsou projektové plány ve skutečnosti tři: jeden pro dimenzi provedení (hierarchická struktura činností), jeden pro dimenzi času (nejlépe síťový diagram, ale občas i seznam milníků nebo úsečkový graf) a jeden pro dimenzi nákladů (finanční rozpočet).

Hierarchická struktura činností je metoda, díky které lze rozdělit projekt na jednotlivé činnosti, které je snazší naplánovat, řídit i kontrolovat. Činnosti jsou spojeny v hierarchickou strukturu, která slouží pro další plánování projektu. Časové i nákladové plány přímo souvisí s hierarchickou strukturou činností a nesmí mít rozdílnou strukturu. Časové a nákladové plány vznikají na základě zkušeností a konzultací projektového manažera s odborníky v dané oblasti, případně pracovníky za danou oblast zodpovědnými. Množství informací a zkušeností, které jsou v tomto kroku projektovému manažerovi k dispozici přímo ovlivňuje kvalitu plánu. Z tohoto důvodu je velmi vhodné evidovat a archivovat plánování i průběh celého projektu.

Součástí plánování projektu je rovněž optimalizace projektu a analýza rizik. Časovou optimalizaci lze nejlépe provádět na základě síťových diagramů popsaných v následující kapitole. Nejčastěji používanou metodou optimalizace je metoda výpočtu kritické cesty, kterou popisuje kapitola 1.1.2.

Řízení projektu a vedení lidí se prolíná celým projektem. Od začátku do konce projektu je nutné pracovníky vést, motivovat i kontrolovat. Různé organizace mají různou organizační strukturu a tu je nutné při řízení projektu respektovat a správně využívat. Nejčastějšími typy organizační struktury jsou *funkční organizační struktura*, *projektová organizační struktura* a *maticová organizační struktura* [4]. Projektový tým běžně vzniká

napříč organizační strukturou a velmi často jsou v projektu zastoupeni i externí pracovníci. Může se jednat profesionální konzultanty, zástupce subdodavatelů nebo naopak zástupce zadavatele projektu. Na členy projektového týmu jsou často kladeny vyšší požadavky, zvláště na ty pracovníky, kteří pracují na více projektech zároveň nebo kromě práce na projektu pracují i na své stále pozici.

Projektové řízení poskytuje množství metodik, jak správně vést členy projektového týmu. Neměli bychom zapomenout poznamenat, že jedním z podstatných požadavků je zajištění dostatečné vzájemné komunikace mezi členy týmu.

Sledování postupu prací na projektu je další řídící prací manažera. Správné monitorování projektu je základním předpokladem pro včasné odhalení zpoždění prací na některé části projektu nebo řešení krizových situací. Základním předpokladem pro přesné monitorování postupu je vhodné rozčlenění projektu a definování takových postupů, které umožní reálně odhadnout postup práce. Kontrolování projektu se nejčastěji provádí prostřednictvím kontrolních schůzek, podáváním zpráv a reportů. Formu a obsah zpráv většinou specifikují směrnice jednotlivých organizací.

S kontrolou postupu prací na projektu rovněž souvisí kontrola čerpání prostředků, i na tuto oblast existují metodiky upravované jednotlivými organizacemi.

V případě detekce zpoždění projektu nebo nutnosti řešení nevyhnutelných problémů se dostává k slovu tzv. *změnové řízení*. Cílem změnového řízení je upravit plán projektu takovým způsobem, aby byl projekt dokončen jen s minimálními změnami trojimperativu.

Vyhodnocení a ukončení projektu je poslední částí projektu. Tato oblast se asi nejvíce liší napříč různými oblastmi ve kterých může být projektové řízení aplikováno. Např. při přípravě veletrhu bude vyhodnocení probíhat až po dnu konání a ukončovací práce budou mimo jiné zahrnovat fakturaci služeb nebo zpracování anket. Při vývoji software je ukončení projektu spojeno s finálním testováním produktu u zákazníka a následným předáním.

Přestože se uvedené příklady liší, existují činnosti, které by rozhodně neměly zůstat opomenuty. Jedná se zvláště o zpětné zhodnocení projektu, kontroly, audity a rovněž zajištění aktualizace know-how společnosti, které usnadní řízení obdobných projektů v budoucnu.

Protože existují různé druhy organizací a projektů, je i náplň jednotlivých oblastí projektového řízení různá. I přes to by měl každý projekt obsahovat všech pět uvedených fází.

1.1.1 Zobrazení struktury projektu

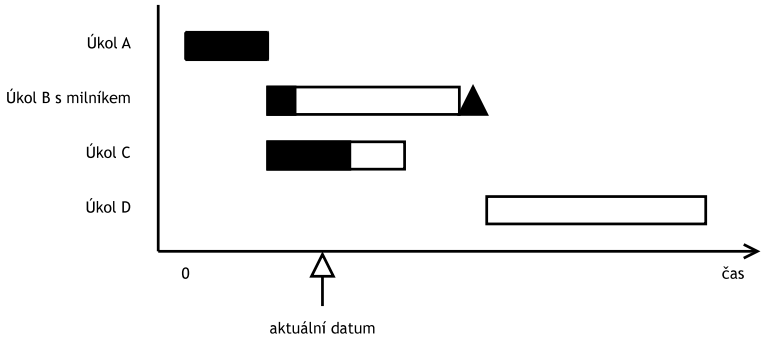
Naplánovaný projekt je vhodné vizualizovat, zobrazit jeho strukturu, časový plán i závislosti úkolů. Dnešní programové vybavení umožňuje zobrazit projekt v různých režimech, které odpovídají nejlépe danému požadavku. Nejčastěji se pro zobrazení časového plánu projektu používají *úsečkové grafy*, *milníky* a *síťové grafy*.

Úsečkové grafy

Úsečkové grafy jsou nejjednodušším způsobem zobrazení časového plánu projektu. Díky své jednoduchosti umožňují získat představu o časovém rozložení činností v rámci projektu. Nejznámějším představitelem těchto grafů jsou tzv. *Ganttovy diagramy* pojmenované po jejich tvůrci, Henri Ganttovi.

Přestože jsou Ganttovy diagramy jednoduché a přehledné, mají jednu zásadní nevýhodu, díky které nejsou prakticky využitelné pro *řízení* projektu. Tyto diagramy sice umožňují

zobrazit průběh jednotlivých úkolů, ale nezobrazují závislosti mezi jednotlivými úkoly. Při zpoždění některého úkolu proto nelze rozeznat, jak toto zpoždění ovlivní další úkoly nebo celý projekt. Ganttův diagram ilustruje obrázek 1.1.



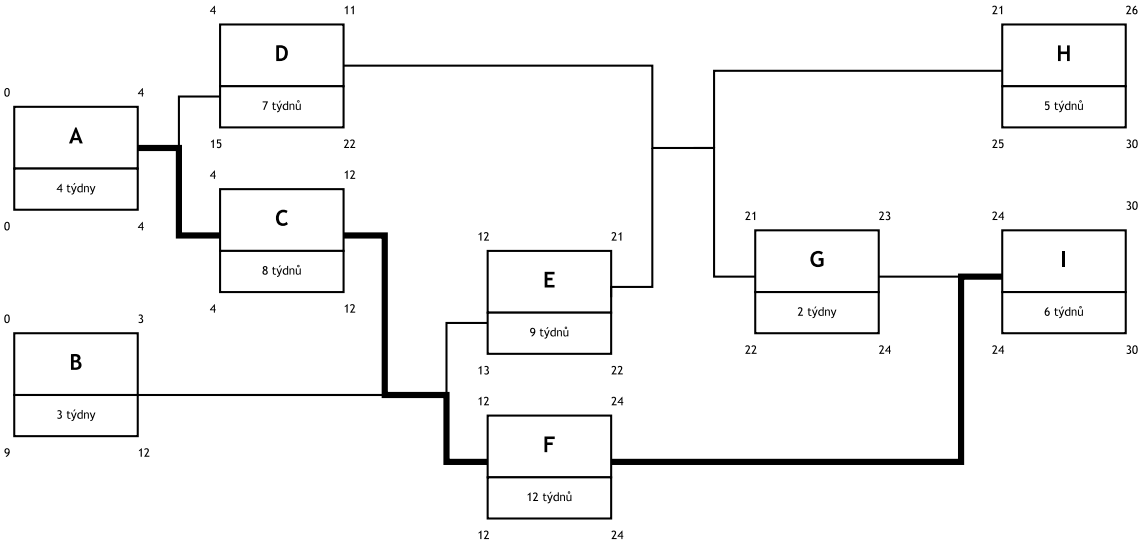
Obrázek 1.1: Ganttův diagram s vyznačeným milníkem

Milníky

Časový plán milníků slouží k zaznamenání několika klíčových událostí projektu, tzv. milníků. Tyto milníky slouží k ověření postupu prací na projektu a často musí být před dalším postupem schváleny. Milníky jsou součástí většiny projektů a k jejich vizualizaci se využívá časová osa. Často bývají kombinovány s úsečkovými grafy.

Síťové grafy

Síťové grafy jsou obecně doporučovanými postupy pro všechny druhy projektů. V projektovém řízení se objevily v padesátých letech dvacátého století a v současné době existuje velké množství jejich variací a kombinací, které různě spojují jejich vlastnosti.



Obrázek 1.2: PERT graf vzorového projektu

Mezi nejběžnější síťové grafy patří síťový graf logického sledu činností (PERT), uzlově orientovaný síťový graf (PDM) a hranově orientovaný síťový graf (ADM)[7]. Graf PERT (zkratka z anglického názvu *Program Evaluation and Review Technique*) byl poprvé využit při vývoji raket pro americkou armádu. Využívá se často v projektech v oblasti výzkumu a vývoje, u nichž je doba trvání nejistá. Činnosti jsou v tomto grafu zobrazeny jako uzly grafu. Naproti tomu hranově orientovaný graf ADM (Arrow diagramming method) zobrazuje činnosti na hraně, přechodu mezi uzly. ADM graf byl aplikován ve stavebním průmyslu, kde lze obvykle dobu provádění každé činnosti ovlivnit. Uzlově orientovaný graf PDM (Precedence diagramming method) je podobný PERT diagramu, ale na rozdíl od něho představují uzly činnost s danou dobou trvání.

Nevýhodou síťových grafů je, že v základní podobě nezobrazují časovou osu a délku trvání jednotlivých úkolů. Tento problém řeší např. graf typu TSTETIL, který kombinuje zobrazení činnosti na hraně s časovou osou.

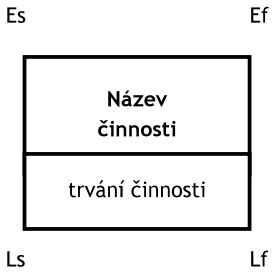
Současná softwarová podpora projektového řízení přinesla větší možnosti v zobrazování síťových grafů a programy často umožňují zobrazit strukturu projektu více způsoby.

1.1.2 Optimalizace projektu

Každý naplánovaný projekt je vhodné optimalizovat. Optimalizace neprobíhá pouze ve fázi plánování projektu, ale je nezbytná při změnovém řízení. Nejčastěji využívanou optimalizací je výpočet kritické cesty projektu a optimalizace přiřazených zdrojů. Základem těchto metod jsou síťové grafy.

Metoda výpočtu kritické cesty (CPM - Critical Path Scheduling) je nejčastěji používanou metodou optimalizace a plánování projektu[5]. Tato metoda umožňuje spočítat minimální čas nutný pro dokončení projektu spolu s časovými údaji o začátku a konci jednotlivých dílčích podprojektů. Kritická cesta projektu reprezentuje posloupnost navazujících úkolů, kdy zdržení jednoho z těchto úkolu má za následek zdržení celého projektu. Proto je nutné těmto úkolům věnovat zvýšenou pozornost a případně k nim přiřadit více zdrojů, např. pracovníků nebo strojů.

Pro zobrazení struktury projektu a závislostí činností je využit PERT graf zobrazený na obrázku 1.2. Čísla zobrazená v diagramu vyznačuje obrázek 1.3, kritická cesta je zobrazena silnější spojovací čarou mezi činnostmi.



Obrázek 1.3: Význam číselných údajů na PERT diagramu

Výpočet kritické cesty

Pro výpočet kritické cesty projektu potřebujeme znát graf projektu a délku trvání jednotlivých činností. K tomu postačuje znát u každé činnosti seznam činností, na kterých je

daná činnost závislá a dále seznam činností, které jsou na dokončení dané činnosti závislé. Obrázek 1.2 zobrazuje jeden jednodušší projekt, který obsahuje devět definovaných činností.

Čísla v rozích každé činnosti označují časové vlastnosti každé činnosti:

- E_S *(Earliest Start)* – označuje nejdříve možný začátek činnosti. Pro první činnost v projektu má hodnotu 0,
- E_F *(Earliest Finish)* – označuje nejdřívější možný konec činnosti,
- L_S *(Latest Start)* – označuje nejzazší možný začátek činnosti aby nebyl narušen časový plán projektu,
- L_F *(Latest Finish)* – označuje nejzazší možný konec činnosti, který nenaruší časový plán projektu.

Výpočet kritické cesty se skládá ze dvou po sobě jdoucích algoritmů:

1. *Earliest Event Time Algorithm* vypočte nejdřívější možný začátek a konec činností, hodnotu E_S a E_F.
2. *Latest Event Time Algorithm* vypočte nejzazší možný začátek a konec činností, hodnoty L_S a L_F.

První algoritmus probíhá od první činnosti v grafu projektu k poslední. Hodnota E_S se rovná maximu z hodnot E_F všech přímých předchůdců počítané činnosti, neboli maximu z nejdřívějšího dokončení všech činností, na kterých činnost závisí. $E_F = E_S + D$, kde D je délka trvání činnosti.

Výpočet algoritmu *Latest Event Time Algorithm* probíhá v opačném směru než algoritmus *Earliest Event Time Algorithm*. Výchozím bodem výpočtu je poslední činnost v grafu a její hodnota E_F.
Poslední činnosti nastavíme hodnotu $L_F = E_F$ a $L_S = L_F - D$. Pro další činnosti v grafu vybíráme vždy minimální hodnotu L_S ze všech následníků počítané činnosti.

Vypočtené hodnoty nám slouží jako základ pro nalezení kritické cesty projektu a rovněž k nalezení rezerv v časovém plánu. Činnosti na kritické cestě jsou takové, pro které platí

$$E_S = L_S, E_F = L_F. \tag{1.1}$$

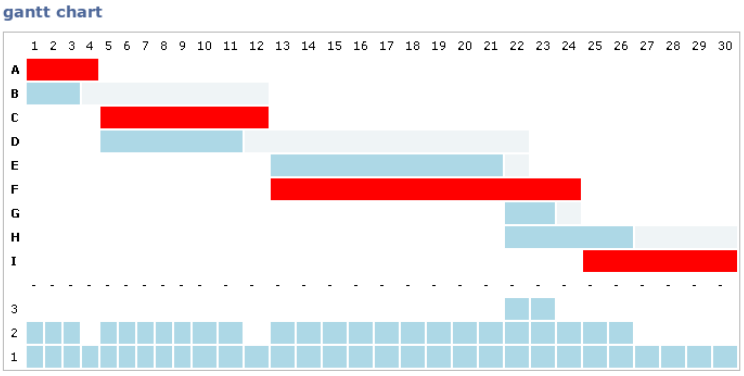
Činnosti, pro které tento vztah neplatí mohou začít kdekoli v rozmezí $\langle E_S, L_S \rangle$ bez vlivu na celkový čas dokončení projektu.

Metoda výpočtu kritické cesty nedává manažerovi projektu pouze informaci o činnostech, které leží na kritické cestě a které se pro včasné dokončení projektu nesmí zpozdit. Z výstupu této metody lze rovněž vypočíst tzv. *polštář* (v anglické literatuře se používá termín *float*), který mohou mít činnosti neležící na kritické cestě. Polštář je hodnota, o kterou se může zpozdit začátek činnosti, aniž by toto zpoždění mělo vliv na následující činnosti nebo celý projekt.

Free float označuje maximální zpoždění začátku činnosti, aniž by toto zpoždění mělo vliv na kteroukoli navazující činnost.

$$FF = L_S - E_S \tag{1.2}$$

Total float je maximální zpoždění začátku činnosti, které neovlivní plánované dokončení projektu, ale ovlivní začátek navazujících činností.



Obrázek 1.4: Ganttův diagram ukázkového příkladu s vyznačeným *polštářem* úkolů a časovým rozložením zdrojů

Obrázek 1.4 ukazuje Ganttův diagram s vyznačenou kritickou cestou, polštářem úkolů a časovým rozložením zdrojů. Obrázek byl vegenerován informačním systémem, který byl implementován jako součást této diplomové práce.

1.1.3 Softwarová podpora projektového řízení

Programové vybavení počítačů, které podporuje projektové řízení je stejně rozsáhlé jako samotné oblasti projektového řízení. Pojem *Project Management Software* zahrnuje mnoho typů progamů, které se zaměřují na podporu plánování, plánování rozpočtu, sledování projektu, dokumentaci projektu nebo alokaci zdrojů.

Project management software lze rozdělit podle způsobu implementace do dvou základních oblastí:

Desktopové aplikace jsou programy spouštěné na počítači každého uživatele. Mohou poskytovat plnohodnotné uživatelské rozhraní s rychlou odezvou. Ke spuštění programu většinou není nutné připojení k internetu nebo počítačové síti.

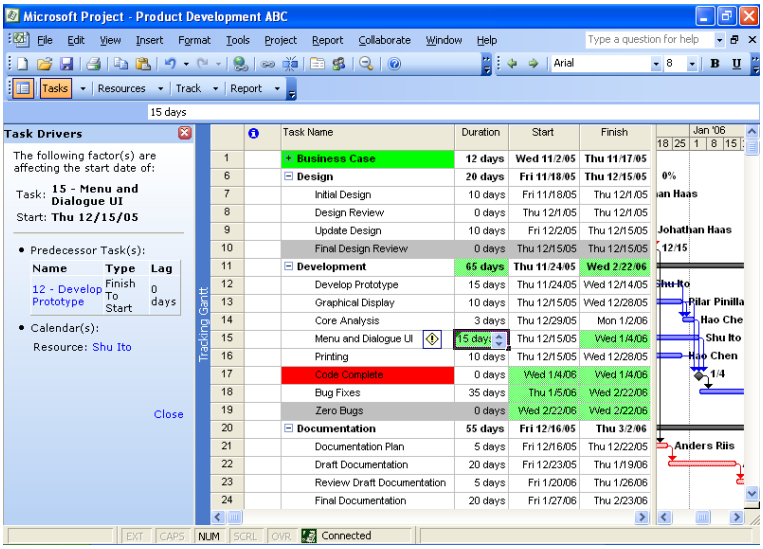
Internetové aplikace vyžadují pro spuštění internetový prohlížeč. Tyto programy mají všechny výhody i nevýhody, které jsou platné pro všechny ostatní internetové aplikace. Mezi hlavní výhody patří

- dostupnost z každého počítače připojeného k internetu,
- nezávislost na operačním systému uživatele,
- nevyžadují instalaci doplňujícího software,
- jsou z principu navržené jako víceuživatelské aplikace,
- všichni uživatelé vždy pracují s aktuální verzí programu.

Typické nevýhody těchto aplikací jsou opět podobné ostatním internetovým aplikacím

- uživatelské rozhraní je omezené internetovým prohlížečem. Toto omezení lze obejít využitím např. technologie Adobe Flash pro implementaci uživatelského rozhraní,
- nedostupnost dat v případě výpadku počítačové sítě,

- obecně pomalejší odezva na akce uživatele.



Obrázek 1.5: Microsoft Project – zdroj www.microsoft.com

Mezi nejznámější desktopové programy pro podporu projektového řízení patří komerční programy Microsoft Project[2] (obrázek 1.5) nebo OmniPlan[3]. Z open-source programů to jsou programy GanttProject[4], Open Workbench[5] nebo součást kancelářského software K-Office, program KPlato[6].
Všechny uvedené programy umožňují rozdělit projekt na dílčí činnosti, definovat jejich závislosti, přiřadit zdroje k úkolům a zaznamenávat postup na projektu.

Internetové aplikace neposkytují takovou podporu pro zobrazení struktury projektu a jednoduchou úpravu grafů, přidávají ale funkce, které tuto nevýhodu vynahrazují. Z komerčních produktů jsou zajímavé aplikace JIRA[7], Basecamp[8] (obrázek 1.6) nebo @Task[9]. Z open-source produktů patří mezi nejznámější dotProject[10] nebo TUTOS[11].

1.2 Týmová spolupráce

Ve všech oblastech lidské činnosti je nutná vzájemná spolupráce a komunikace mezi lidmi. V rámci jednoho projektu je tato spolupráce ještě důležitější. Softwarové nástroje, které jsou určené pro podporu týmové spolupráce, se nazývají *groupware* nebo *collaboration software*.

Obecně můžeme za nástroje určené k podpoře týmové spolupráce považovat například telefon, flipchart nebo konferenční místnost. Tedy vše, co umožňuje lidem efektivněji spo-

[2] http://office.microsoft.com/en-us/project
[3] http://www.omnigroup.com/applications/omniplan
[4] http://ganttproject.biz
[5] http://www.openworkbench.org
[6] http://www.koffice.org/kplato
[7] http://www.atlassian.com/software/jira
[8] http://basecamphq.com
[9] http://www.attask.com
[10] http://www.dotproject.net
[11] http://www.tutos.org

lupracovat a komunikovat na společném díle. V oblasti programového vybavení je rozsah těchto nástrojů obdobně široký. Z pohledu úrovně spolupráce můžeme groupware aplikace rozdělit do čtyř kategorií: nástroje pro elektronickou komunikaci, elektronické konference, nástroje pro řízení týmové spolupráce a nástroje pro spolupráci při vytváření obsahu.

Nástroje pro přímou komunikaci slouží k přímé komunikaci mezi dvěma uživateli, k zasílání zpráv, dokumentů nebo dat obecně. Do této oblasti řadíme e-mail, programy pro rychlé zasílání správ (např. MSN Messanger, ICQ, GTalk a podobně).

Nástroje pro elektronické konference slouží k předávání informací a dat v rámci skupiny uživatelů. Jedná se o internetová fóra, chatovací systémy, telefonní a video konference.

Nástroje pro řízení týmové spolupráce se používají k více či méně organizované spolupráci nebo komunikaci ve skupinách lidí. Jedná se například o sdílené elektronické kalendáře, nástroje pro podporu řízení projektů, workflow systémy (např. help-desk aplikace) a rovněž tzv. social software.

Nástroje pro spolupráci při vytváření obsahu, jejichž nejznámějším představitelem současnosti je otevřená encyklopedie Wikipedia, slouží k zapojení co největšího počtu lidí do tvorby nejrůznějších dokumentů. Dalšími představiteli této kategorie jsou například on-line textové nebo tabulkové editory, systémy pro správu poznámek nebo nástroje pro správu verzí (CVS, SVN).

Obrázek 1.6: Basecamp společnosti 37signals, LLC představuje špičku mezi internetovými aplikacemi

Pro efektivní spolupráci a komunikaci je nutné nejrůznější groupware aplikace integrovat do jednoho systému, případně zajistit vzájemné provázání [2]. Příkladem může být aplikace Google Docs and Spreadsheets[12], která integruje nástroj pro vytváření obsahu, nástroj pro komentování obsahu a rovněž zasílání e-mail zpráv.

Druhou, a v určitých oblastech mnohem důležitější vlastností groupware aplikací je kvalitní uživatelské rozhraní. Úspěch všech groupware aplikací je založen na co největším využívání aplikace uživateli, a ovládání aplikace hraje v této oblasti velkou roli.

[12] `http://docs.google.com`

1.3 IAESTE České republiky

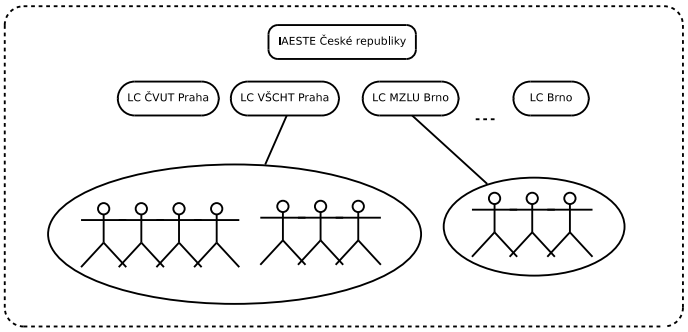
Jak bylo uvedeno výše, je oblast projektového řízení velmi rozsáhlá. Obecné systémy pro podporu projektového řízení jsou často příliš složité a komplikované pro uživatele. Cílem tohoto projektu je vytvořit takový systém, který bude využitelný v menších organizacích s různými typy projektů. Jako vzorovou organizaci byla vybrána nezisková studentská organizace IAESTE České republiky. Tato kapitola popisuje strukturu organizace a typy projektů.

Poslání IAESTE ČR se skládá ze tří rovnocenných oblastí: zajišťování mezinárodního výměnného programu odborných stáží, organizování personalistických projektů zaměřených na studenty a absolventy vysokých škol a odborný rozvoj vlastních členů.

1.3.1 Organizační struktura IAESTE ČR

IAESTE České republiky je organizace zastřešující osm lokálních centrech působících na českých univerzitách. Lokální centra mají vlastní právní subjektivitu, ale musí se řídit rozhodnutími Senátu nebo Dozorčí rady IAESTE ČR. Členy IAESTE ČR jsou členové všech Lokálních center. Členství se vztahuje vždy k jednomu lokálnímu centru, viz obrázek 1.7.

Většina činností v IAESTE je tedy vázána k lokálním centrům. Kromě činností v rámci lokálního centra probíhá několik projektů, které jsou koordinované napříč celým IAESTE ČR a do projektů se zapojují členové z více center zároveň. Koordinaci těchto projektů zajišťuje Národní centrum, neboli IAESTE České republiky.



Obrázek 1.7: Organizační struktura IAESTE ČR

1.3.2 Organizační struktura lokálního centra

Lokální centra se dají svojí strukturou rozdělit do dvou skupin, která můžeme nazvat *malá* a *velká centra*. Velká centra, v současnosti LC ČVUT a LC VŠCHT Praha, mají řádově desítky členů, malá centra do desíti členů. Tento rozdíl v členské základně se pochopitelně projevuje na způsobu vedení centra, projektů, i organizační struktuře centra.

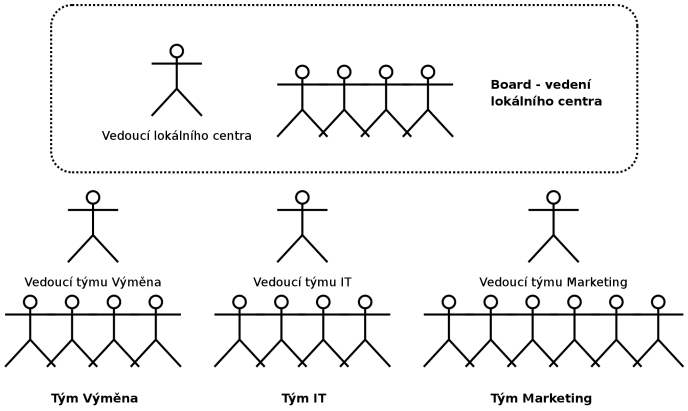
Velká lokání centra

Velká centra mají hierarchickou strukturu s pevně daným rozdělením členů do týmů a rovněž jasně danou pozicí v rámci centra (obrázek 1.8). Členové jsou rozděleni do jednotlivých týmů, které pokrývají různé činnosti centra. Jedná se například o tým Marketingu,

Výměny, IT, Public Relations, Human Resources. Každý z těchto týmů má svoje projekty, na kterých ale mohou pracovat i členové z jiných týmů nebo jiných lokálních center. Vedoucí týmu stojí v čele svého týmu a je zodpovědný za jeho činnost, to znamená za kontrolu, případně vedení projektů, kontrolování a hodnocení práce členů týmu a informování vedení centra o činnosti týmu.

Vedení centra (board) se skládá z několika členů, většinou vedoucích týmů a je zodpovědné za chod celého centra. Součástí vedení centra je i vedoucí lokálního centra, který je statutárním zástupcem lokálního centra.

Na všech úrovních lokálního centra probíhají pravidelné schůze, ze kterých se pořizují zápisy a kontroluje se plnění úkolů. Tyto zápisy musí být dostupné všem členům týmu, případně projektu a rovněž vedení centra.



Obrázek 1.8: Organizační struktura velkého lokálního centra

Malá lokální centra

Malá lokální centra se liší od velkých jak počtem členů, tak organizací práce. Ve vedení centra opět stojí board a vedoucí centra. V malých lokálních centrech už ale neexistuje pevné rozdělení do týmů. Týmy jako u velkých center zde neexistují, jsou zde pouze členové zodpovědní za určitou oblast a projekty, na kterých členové pracují.

1.3.3 Popis projektů a činností IAESTE ČR a lokálních center

V organizacích typu IAESTE probíhá celoročně velké množství nejrůznějších projektů. Od dílčích úkolů přes projekty, do kterých se zapojují i desítky lidí, po dlouhodobé činnosti, které už nelze považovat za projekty. Práci v IAESTE můžeme rozdělit do dvou oblastí:

Projekty

V IAESTE probíhá velké množství činností, které lze podle definice z kapitoly 1.1 jednoznačně označit jako projekty. Jedná se například o projekty Katalog pracovních příležitostí, Veletrh pracovních příležitostí nebo Průvodce prváka. Na všechny tyto projekty lze uplatnit metody a postupy projektového řízení.

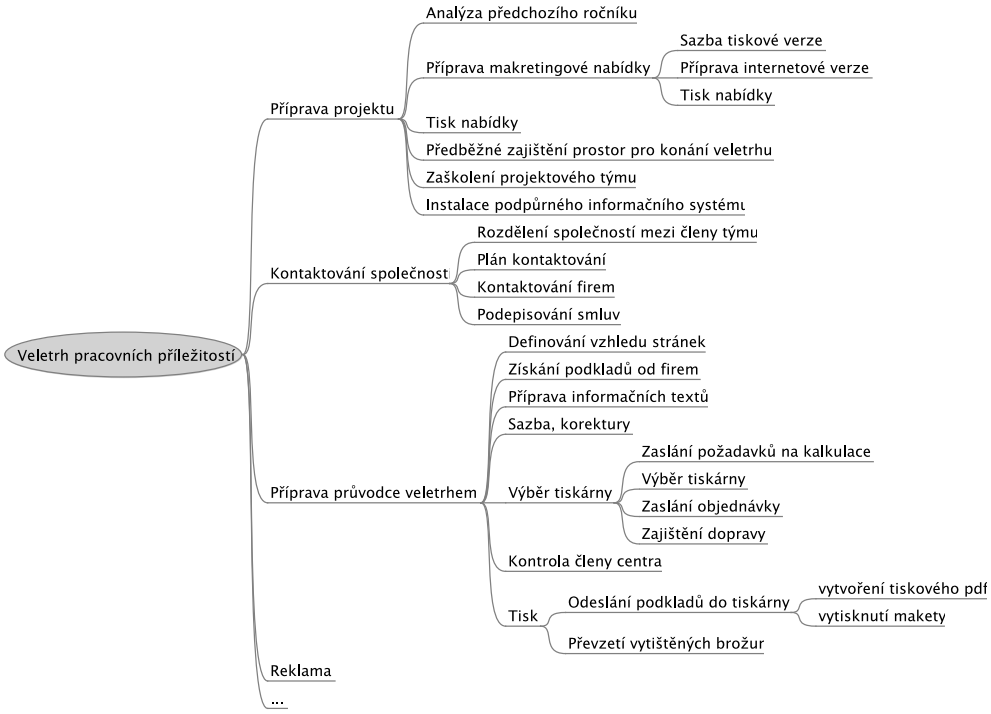
Dlouhodobé činnosti

Kromě projektů existují v IAESTE další aktivity, které z nejrůznějších důvodů nelze považovat za projekty. Nejčastějším důvodem je časová neomezenost těchto aktivit. Do této oblasti spadá například vedení lokálního centra, koordinace mezinárodního výměnného programu na úrovni lokálního i národního centra, propagace IAESTE nebo správa internetových stránek. Pro všechny tyto činnosti je potřeba v budoucím systému zajistit podporu a návaznost na funkce projektového řízení, které lze uplatnit na části této činnosti. Např. v oblasti propagace IAESTE lze za dílčí projekt považovat vytvoření nových plakátů a článků.

Pro větší názornost uvedeme dva příklady činností každého lokálního centra IAESTE České republiky. Jedná se o dvě odlišné činnosti, ať již rozsahem nebo výsledným *produktem*.

Veletrh pracovních příležitostí je stěžejní komerční projekt lokálních center. Vlastní veletrh pracovních příležitostí pořádají téměř všechna lokální centra. Jedná se o veletrh, který probíhá v jeden den v prostorách vysoké školy. Veletrhu se účastní deset až více než sto firem. Každá firma má zajištěný stánek, prezentaci v posluchárně a vlastní stránku v tištěné brožuře Průvodce veletrhem. Součástí projektu je rovněž reklamní kampaň i získávání zpětné vazby od zástupců firem a návštěvníků veletrhu.

Z pohledu projektového řízení se jedná o klasický projekt, který má přesně dané zadání, projektového manažera, tým pracovníků a je přesně časově omezený dnem konání veletrhu.



Obrázek 1.9: Ukázka hierarchické struktury činností Veletrhu pracovních příležitostí

Projekt se skládá z několika podprojektů a spousty dalších činností, které na sebe navazují, případně probíhají současně. Hierarchickou strukturu činností zobrazuje obrázek 1.9.

Z obrázku, který nezobrazuje celý projekt, vidíme, že přestože se jedná z pohledu projektového řízení o *malý projekt*, obsahuje už v této zjednodušené verzi spoustu činností, které na sebe bezesporu musí navazovat a je nutné kontrolovat jejich plnění, aby byl projekt realizovaný v daném termínu. Poznamenejme, že termín Veletrhu je prakticky nezměnitelný.

Koordinace výměnného programu se od přípravy Veletrhu pracovních příležitostí liší v mnoha oblastech. Zatímco Veletrh je komerční projekt, jehož cílem je, kromě získání zkušeností z realizace zajímavého projektu, zisk nutný pro pokrytí nákladů na provoz lokálního centra a zvláště mezinárodního výměnného programu. Výsledkem dobře zvládnuté koordinace výměnného programu jsou „pouze“ spokojení účastníci odborné stáže – čeští nebo zahraniční studenti a jejich zaměstnavatelé.

Koordinace výměnného programu není projekt. Je to dlouhodobá činnost, která má během roku několik milníků (např. termín odeslání požadavků na zahraniční praxe) a jednotlivé dílčí činnosti se vzájemně překrývají. Na tyto dílčí činnosti už lze pohlížet jako na malé projekty, které jsou vzájemně provázané. Z nejdůležitějších můžeme jmenovat zorganizování konkurzu, vyhledávání praktikantských pozic pro zahraniční studenty, administrativa spojená s českými nebo zahraničními studenty a plánování programu pro zahraniční studenty (výlety, exkurze, párty, ...).

Podobný přístup, jako uvedená koordinace výměnného programu, vyžadují další činnosti v rámci IAESTE. Jmenujme např. vedení lokálního centra nebo práci týmu public relations, který vytváří podklady pro prezentaci organizace IAESTE. Z uvedených příkladů mimojiné vyplývá, že na většinu činností lze v IAESTE České republiky použít metody projektového řízení.

1.3.4 Aplikace používané v IAESTE ČR

Poslední část první kapitoly představí současný stav využívání informačních technologií v IAESTE České republiky. Toto představení má vliv na zbývají část práce, protože přímo ovlivňuje požadavky na nový systém.

IAESTE ČR v současnosti využívá velké množství desktopových aplikací a rovněž několik internetových aplikací. Je to dáno strukturou organizace, historickým vývojem i způsobem práce (mnoho členů využívá k práci na projektech počítače na fakultách nebo v knihovnách). Nejčastěji využívané programy jsou e-mailoví klienti (MS Outlook, MS Outlook Express, Mozilla Thunderbird), internetové prohlížeče (MS Internet Explorer, Mozilla Firefox, Opera), kancelářské programy (MS Office, OpenOffice.org) a grafické programy. Internetové aplikace popisují následující kapitoly.

Intranet IAESTE ČR

Je základním informačním systémem, který obsahuje nejdůležitější data IAESTE ČR. Skládá se z několika částí:

Company Index je nástroj pro správu kontaktů. Obsahuje seznam firem, se kterými IAESTE spolupracuje, eviduje komunikaci s firmami a účast na projektech. V Company Indexu je možné vytvořit projekty, např. Veletrh iKariéra 2007 ČVUT a přiřadit firmám účast či neúčast v tomto projektu, evidovat komunikaci s firmou v rámci tohoto projektu. K projektu je možné nastavit uzávěrku, koordinátora, doplňující poznámky a odkaz na adresář projektu na Sharedisku. Na Company Index se vážou další služby Intranetu.

Faktury je služba, která umožňuje k firmám v Company Indexu vytvořit faktury. Systém zajišťuje správné číslování faktur a je přístupný pouze pro vedoucí center.

Úkoly umožňují označit zápis komunikace v Company Indexu jako úkol, např. pro příští kontaktování firmy. Tuto funkci lze využít jako velmi jednoduchý groupware pro komunikaci při práci na projektu.

Seznam členů – Intranet obsahuje seznam a kontakty na všechny členy IAESTE ČR rozdělený podle lokálních center. Jedná se o část Company Indexu.

Sharedisk je sdílený diskový prostor pomocí technologie WEB-DAV. Administrátor může definovat přístupová práva do jednotlivých adresářů.

Nástěnka je webové fórum, které slouží pro různá oznámení, novinky a případně návody.

Podpora projektu Katalog je rozhraní pro registraci firem do Katalogu iKariéra a online tvorbu obsahu stránky zástupcem firmy.

Podpora projektu Výměna umožňuje prostřednictvím veřejných formulářů na internetových stránkách IAESTE registraci studentů na konkurzy, výběr preferovaných praxí pro burzu praxí, registrace přijíždějících studentů. Tato část Intranetu bude v budoucnu nahrazena novou, pravděpodobně externí, aplikací.

E-mailové skupiny jsou skupinové e-mailové adresy do kterých je možné se přihlásit nebo odhlásit prostřednictvím Intranetu. Intranet obsahuje rovněž kompletní archív těchto skupin.

Interní fotogalerie generuje z obrázků nahraných ve zvláštní složce na serveru jednoduchou fotogalerie. V plánu je přechod na vybraný open-source systém, který umožní komfortnější prohlížení obrázků.

E-mail

E-mailové účty v doméně iaeste.cz jsou uloženy na IMAP serveru. Členové využívají ke čtení pošty různé e-mailové klienty. Nejčastěji webové rozhraní Horde, SquirrelMail nebo Gmail, dále desktopové aplikace Mozilla Thunderbird, Microsoft Outlook.

IAESTEpedia

Wiki IAESTE ČR. Systém pro předávání zkušeností a znalostí založený na open-source softwaru Tikiwiki.

Portál iKariéra.cz

Portál iKariéra.cz obsahuje vlastní databázi firem, která je odlišná od Company Indexu a rovněž obsahuje jiné přihlašovací účty. Databáze portálu se využívá pro prezentaci firem v reklamní části serveru iKariéra.cz, registrace firem na veletrhy pracovních příležitostí, vyplňování textů do průvodců veletrhem a evidenci požadavků firem.

1.4 Software as a Service

Software as a Service (SaaS) je model poskytování software formou pronájmu, tedy formou služby. Zákazníci si v tomto modelu najímají možnost využívat internetovou aplikaci běžící na vzdáleném internetovém serveru. Model SaaS předpokládá, že jednu instalaci aplikace na serveru využívá více klientů.

Cílem Saas je snížit náklady na provoz informačních systémů ve firmách. Současný stav využívání informačních technologií vyžaduje po společnostech správu vlastních serverů a jejich obsluhu. Využitím SaaS se přenáší tato zodpovědnost na poskytovatele služby.

Kapitola 2

Specifikace požadavků

Předchozí kapitola nás uvedla do problematiky projektového řízení a týmové spolupráce a vysvětlila vzájemnou provázanost těchto oblastí. Víme, jaké činnosti představuje projektové řízení i způsoby, jak jednotlivé oblasti podporují informační technologie. Závěr kapitoly byl věnován vzorové studentské organizaci IAESTE České republiky, její organizační struktuře a projektům, které tato organizace pořádá.

Tato kapitola specifikuje požadavky na systém, který si klade za cíl podpořit, usnadnit a zlepšit projektové řízení ve vybrané organizaci.

2.1 Projektové řízení v IAESTE České republiky

Všechny projekty, které jsou řízené v rámci IAESTE ČR, můžeme považovat z pohledu projektového řízení za malé. Trvají maximálně několik měsíců a mají výrazně omezené zdroje. Z tohoto důvodu nejsou některé metody projektového řízení natolik podstatné, aby je bylo nutné zahrnovat do specifikace požadavků nebo pro ně vytvářet nové aplikace. Následující seznam popisuje oblasti projektového řízení a hodnotí požadavky na softwarové nástroje, které budou v prostředí IAESTE ČR dostatečně danou oblast pokrývat.

Definice cílů projektu nevyžaduje žádné speciální programové vybavení. Pro analýzu projektu nebo sestavení zadání plně postačuje textový editor a podpora sdílení souborů. Analýza projektu často vyžaduje získání zkušeností z obdobných projektů organizovaných v minulosti, proto je vhodné zajistit přístup ke zprávám předchozích projektů, uchovat kontakty na projektové manažery i členy projektových týmů a rovněž využívat některý z programů podporující znalostní management – v případě IAESTE ČR se jedná o aplikaci IAESTEpedia – 1.3.4.

Plánování projektu je oblast, na kterou se softwarová podpora projektového řízení nejvíce zaměřuje, viz. kapitola 1.1.3.
Hierarchickou strukturu činností lze snadno navrhnout s tužkou a papírem. Ze softwarových nástrojů stojí za pozornost open-source aplikace FreeMind[1], určená pro tzv. *mapování myšlenek*[2]. FreeMind umožňuje přehledně vytvářet a měnit hierarchii činností.

Plánovací nástroje, využívající Ganttovy nebo síťové grafy jsou nezbytnou součástí každého systému pro podporu řízení projektů. Proje je vhodné je implementovat a uplatnit

[1] `http://freemind.sourceforge.net`
[2] `http://en.wikipedia.org/wiki/Mind_map`

nejen při plánování projektu, ale rovněž při kontrole projektu a změnovém řízení. Definovaní činností a jejich závislostí vždy vede k hlubšímu porozumění projektu, navíc umožňuje aplikovat metodu kritické cesty.

Na síťové grafy navazuje *oblast optimalizace* projektu a přidělení zdrojů k činnostem. Tato oblast je v prostředí IAESTE velmi problematická. Členové nejsou v IAESTE ČR zaměstnáni a projektům se věnují ve svém volném čase. Nemají proto, na rozdíl od běžných zaměstnanců, pevnou pracovní dobu. Metody určené pro optimalizaci projektu jsou naopak založeny na předpokladu relativně přesně počítatelného *výkonu* zdrojů.

Přidělení pracovníků k projektům má ale další přínos, který je nejvíce patrný v neziskových, dobrovolnických organizacích, kde pracovníci (nazývejme je přesněji členové) nemají za odvedenou práci plat. Odměny jsou většinou nefinanční, ať už formou účasti na nejrůznějších akcích, možnosti vzdělávání nebo dobrý pocit ze smysluplné práce. V těchto organizacích je více než kde jinde nutné zajistit transparentní systém hodnocení. Přesné a veřejné přiřazení členů k jednotlivým činnostem a úkolům může jedině prospět k přesnějšímu a odpovědnějšímu odměňování. Uložení všech dat v jednom systému dále umožní snadno vyhledat aktivitu člena a tím přesněji a odpovědněji každého člena hodnotit.

Při plánování projektu je vhodné čerpat informace z průběhu obdobných předchozích projektů. Evidence a archivace průběhu celého projektu se proto může stát zdrojem informací pro budoucí projektové manažery.

Plánování rozpočtu nevyžaduje žádné speciální informační systémy. Pro tento nezbytný krok je ideálním nástrojem tabulkový procesor Microsoft Excel, případně OpenOffice.org Calc. Obdobně jako u analýzy projektu je potřeba zajistit sdílení souborů mezi členy týmu i vedením lokálního centra.

Řízení projektu a vedení lidí částečně popsaly předchozí odstavce. Pro vedení lidí je ale velmi důležitá komunikace mezi vedením projektu a řadovými členy, informování o změnách, důležitých milnících projektu nebo termínech pracovních schůzek. Tuto oblast mají za úkol pokrýt nástroje podpory týmové spolupráce, které byly popsány v kapitole 1.2 a požadavky specifikuje kapitola 2.4.

Sledování postupu prací na projektu představuje hlavně kontrolu práce členů a případné úpravy projektů. Součástí je ale rovněž informování zadavatelů projektu nebo vedení společnosti o průběhu projektu. K tomuto účelu dostatečně slouží elektronická pošta nebo sdílení dokumentů. Vytvoření systému, který by zadané zprávy přehledně evidoval v čase a umožnil zpětnou vazbu od členů by zajisté přineslo výhody a opět zvýšilo průhlednost celého vedení projektu.

Vyhodnocení a ukončení projektu spočívá ve formálním zakončení projektu, provedení kontrol (např. financování) a hodnocení členů. Projektový manažer i členové týmu by měli podle svých zkušeností s projektem aktualizovat know-how společnosti. IAESTE České republiky využívá pro tyto účely IAESTEpedii popsanou v kapitole 1.3.4. Je potřeba ale zdůraznit rozdíl mezi zhodnocením jednoho projektu a předáním informací obecně. Konkrétní projekt by měl zůstat zhodnocený v rámci informačního systému a obecné informace naopak v IAESTEpedii. Pouze tímto způsobem lze zajistit, že informace specifické pro průběh jednoho projektu nebudou v aplikaci typu wiki přepsány.

2.2 Požadavky na strukturu projektu

Předchozí kapitola zhodnotila požadavky na celé projektové řízení v organizaci IAESTE ČR. Vyplývá z ní, mimojiné, že nejpodstatnější částí, kterou je nutné implementovat je oblast definování a strukturování projektu. Ostatní požadavky na nový systém jsou z oblasti podpory týmové spolupráce, která bude popsána dále.

V rámci lokálního centra probíhá většinou více projektů zároveň, některé projekty mohou být připravované nebo už uzavřené. Je proto vhodné zajistit jednoduchou a přehlednou organizaci projektů. U každého projektu proto potřebujeme evidovat:

název a popis projektu ,

stav projektu – projekt se může nacházet v několika stavech: aktivní (právě probíhá), zrušený a ukončený (archivovaný),

kategorie projektu – pro snadné dohledání projektu aktivního nebo uzavřeného projektu je nutné vhodným způsobem zařadit projekty do odpovídajících kategorií,

manažer projektu – člen centra nebo organizace, který je za daný projekt zodpovědný. Jeden projekt může mít pouze jednoho vedoucího projektu,

realizační tým – seznam členů, kteří na projektu pracují. Na projektu mohou pracovat členové z různých center.

Definovaný projekt je potřeba dále rozčlenit na podprojekty. Univerzálním přístupem je umožnění členit projekty na neomezenou hloubku podprojektů. Tento přístup ale přináší nepřiměřené zvyšování složitosti ovládání internetové aplikace, jak ukazují některé open-source aplikace, např. kvalitní, ale uživatelsky velmi nepřívětivý systém TUTOS.
Vzhledem ke struktuře a typům projektů, které v IAESTE probíhají, se možnost členit projekt na neomezenou hloubku podprojektů jeví jako uživatelsky příliš komplikované řešení. Proto je výhodnější zavést pouze jednu úroveň podprojektů a další úroveň (dílčí úkoly) přenést do oblasti podpory týmové spolupráce. Pro správné strukturování projektu a případnou optimalizaci je potřeba evidovat:

název podprojektu ,

zodpovědný člen týmu – podobně jako u celého projektu, i za podprojekt musí být zodpovědný pouze jeden člen týmu. Tímto přesným definováním se zamezí problémům s odpovědností za jednotlivé úkoly,

stav podprojektu – v procentech určená úroveň odvedené práce na projektu. Při 100% je podprojekt označen za dokočený. Určení stavu, rozpracování, podprojektu je obecně jednou z nejčastějších komplikací projektového řízení. Často nelze přesně určit, kolik práce již bylo odvedeno, proto je třeba tento údaj brát jako čistě orientační,

závislosti podprojektu – seznam podprojektů, na kterých je daný podprojekt závislý. Tento seznam slouží k vygenerování síťového grafu a případné optimalizaci projektu.

Je nutné si uvědomit, že uvedené požadavky na strukturu projektu představují nezbytné minumum informací, které je potřeba evidovat pro podporu projektového řízení. Požadovat

po uživateli pouze nezbytné minimum informací je základní přístup, který se prolíná celým navrhovaným systémem. Tento přístup byl zvolen z několika důvodů. Ten nejdůležitější je, že s navrhovaným systémem nebudou pracovat profesionální projektoví manažeři, ale studenti, kteří se s danou problematikou teprve seznamují. Dále nesmíme zapomínat, že členská základna studentských organizací je velmi proměnlivá a všechni noví členové jsou při vstupu do organizace nuceni využívat nové informační systémy. Jednoduchost proto není v tomto případě nevýhodou, ale naopak výhodou.

2.3 Požadavky na organizační strukturu a přístupová práva

V rámci každého informačního systému je nutné specifikovat uživatelské role a přístupová práva do různých částí systému. Požadavky na hierarchickou strukturu organizace a uživatelských účtů jsou u každé organizace různé. Větší firmy většinou dodržují přísně hierarchické uspořádání, menší organizace naproti tomu využívají méně formální rozložení, které umožňuje spolupracovat na projektech napříč klasickou firemní hierarchií. Vzorová organizace, IAESTE České republiky, je vhodným příkladem tohoto přístupu.

Z popisu organizační struktury organizace v kapitole 1.3.1 a projektů v kapitole 1.3.3 vyplývají následující požadavky:

- organizace je rozdělena na samostatné jednotky, tzv. lokální centra, která spravují uživatelské účty, vytvářejí a řídí projekty,
- členové přísluší vždy k jednomu lokálnímu centru,
- členové mohou být členy různých týmů, jeden člen může být součástí více týmů zároveň.

Za projekt je organizačně zodpovědné vždy konkrétní lokální centrum. Přesto mohou na projektu spolupracovat členové ostatních lokálních center. Tato spolupráce může být buď přímá nebo nepřímá.
Přímá spolupráce spočívá v aktivním zapojení do projektu, typicky se takový člen účastní porad, je zodpovědný za úkoly a má svou pozici v projektovém týmu.
Nepřímá účast je taková, kdy člen pouze dočasně pomáhá na projektu. Tato pomoc může být různá. V projektu Veletrh pracovních příležitostí představeném v kapitole 1.3.3 velmi často pomáhají členové lokálního centra při kontrole grafických podkladů před jejich odesláním do tisku. V současnosti nalezené chyby posílají e-mailem nebo prostřednictím ICQ členovi týmu, který grafické podklady připravuje. Tento přístup sice poskytuje velmi rychlou zpětnou vazbu, ale postrádá důležitý faktor evidování takovýchto požadavků. Proto je základním požadavkem na definování přístupových práv možnost zpřístupnit některé oblasti projektu také členům lokálního centra, kteří se nepodílejí přímo na projektu.

2.4 Požadavky na podporu týmové spolupráce

Systém, který podpoří systematický přístup k řízení projektů bude nesporným přínosem pro kteroukoli organizaci, která organizuje více projektů zároveň. Vhodným doplňkem k této funkcionalitě je podpora týmové spolupráce. Jak by měla tato podpora vypadat? V kapitole 1.2 jsme poznali rozsah pojmu groupware, i to, že v některých oblastech se přinejmenším prolíná s projektovým řízením.

Každá organizace je jedinečná. Firemní kultura ovlivňuje nejen osobní vztahy na pracovišti, ale rovněž využívání informačních technologií, ať už se jedná o psaní e-mailů, využívání programů pro rychlé zasílání zpráv nebo internetové telefonie. Specifikovat požadavky na systém, který bude vyhovovat každé organizaci je proto stejně problematické jako specifikovat požadavky na řízení projektů. Protože je IAESTE České republiky nezisková studentská organizace, jsou vztahy mezi všemi členy napříč lokálními centry velmi neoficiální a neexistují regulace komunikace mezi členy. Z toho plyne, že podpora týmové spolupráce by neměla komunikaci žádným způsobem omezovat, ale naopak podporovat.

Základními požadavky na týmovou spolupráci je proto zajištění *efektivní zpětné vazby* a zajištění *transparentnosti*. Druhý požadavek vyplývá z předpokladu, že maximálně otevřená společnost podporuje aktivitu členů – pokud jako vedoucí týmu nebo projektu máme přehled o tom, co který člen dělá a jak se zapojuje, je pro nás snažší správně členy odměňovat nebo jim přidělovat úkoly. Výhoda z toho plyne rovněž pro řadové členy, kteří mají přehled o tom, jak pracují jejich kolegové a proč jsou lépe hodnocení.

A jakým způsobem lze zajistit tyto klíčové požadavky? Zpětnou vazbu nejlépe získáme tak, že umožníme uživatelům systému přímo reagovat na naše podněty. Nástroj, který toto umožní je komentář. Mělo by být možné komentovat všechny prvky v systému, u kterých je to vhodné, např. sdílené soubory nebo zprávy.

K zajištění transparentnosti nejlépe poslouží otevření všech částí systému všem členům týmu, i když někdy pouze pro čtení. Znamená to zobrazení všech úkolů, které na projektu jsou a také toho, kdo je za který úkol zodpovědný. Přesně definovanou zodpovědností za dílčí úkoly zajistíme osobní zodpovědnost členů týmu za dokončení úkolu. Tímto požadavkem jsme se dostali zpět k projektovému řízení.

Další nástroje, které podporují týmovou spolupráci jsou nástroje pro společné vytváření obsahu. Jedná se o online textové editory a podporu sdílení souborů, opět s funkcí komentářů. Nástroje, které jsme pro náš systém vybrali jsou popsány v kapitole 3.1.

2.5 Požadavky na uživatelské rozhraní aplikace

Jak bylo uvedeno v kapitole 1.2, je úspěšnost všech groupware aplikací založena na množství uživatelů, kteří aplikaci *aktivně* využívají. Toho lze docílit buď formou organizačního nařízení, nebo vytvořením aplikace, která přinese uživatelům takový přínos, že nebude nutné používání aplikace vynucovat. Vedle funkcí systému hraje důležitou roli grafické uživatelské rozhraní aplikace (GUI).

Omezujícím požadavkem na uživatelské rozhraní aplikace je přístupnost k aplikaci prostřednictvím internetového prohlížeče z libovolného počítače připojenému k Internetu, bez požadavků na další doinstalovaný software, viz kapitola 2.6.

Základní požadavky na GUI systému lze charakterizovat třemi slovy:

jednoduchost spočívá v zobrazení pouze nezbytných funkcí, které uživatel ke své práci potřebuje,

přístupnost znamená prezentování informací v takové formě, aby byly dostupné i hendikepovaným uživatelům. Při zobrazení dat prostřednictvím internetového prohlížeče je vyžadováno použití dostatečně kontrastních barev, logického členění dokumentu a zejména oddělení informací spojených s formátováním dokumentu od vlastního obsahu dokumentu. Toho lze dosáhnout použitím kaskádovacích stylů (CSS) pro grafický popis dokumentu nebo generováním různých výstupních verzí téhož dokumentu,

přehlednost odpovídá vhodně zvolenému rozložení informací a ovládacích prvků na stránce a použití doplňujících barev a grafiky.

2.6 Technické požadavky a omezení

Na každý informační systém jsou kladena nejrůznější technická omezení. Nejinak je tomu i v rámci tohoto projektu. Základním požadavkem je, aby nový systém mohl být provozován při využití stávajícího technického a programového vybavení.

2.6.1 Na straně klienta

Protože uživatelé využívají pro přístup k aplikacím IAESTE České republiky počítače nejrůznějších konfigurací, existují omezení na výstupní formát dat, které budou odesílány klientskému internetovému prohlížeči. Základním požadavkem je tedy korektní zobrazení a chování aplikace ve „standardních“ internetových prohlížečích Microsoft Internet Explorer, Mozilla Firefox a Opera v jejich posledních verzích používaných na operačních systémech Microsoft Windows, GNU/Linux a Mac OS X. Aplikace musí být použitelná i při vypnuté podpoře JavaScriptu.

2.6.2 Na straně serveru

Oproti požadavkům na klientskou část aplikace jsou omezení, která jsou kladená na serverovou část aplikace, mnohem větší. Požadavky vychází z aktuálně provozovaných serverových operačních systémů v IAESTE České republiky.

Aplikace musí splňovat tyto omezení:

- operační systém: GNU/Linux,
- databázový systém: PostgreSQL nebo MySQL,
- web server: Apache nebo Apache Tomcat,
- programovací jazyk: Java, PHP 5 nebo jakýkoli jiný jazyk, který je provozovatelný na výše uvedené konfiguraci.

Kapitola 3

Analýza požadavků

Obsah této kapitoly vychází z obecných požadavků uvedených v předchozí kapitole. Navrhuje konkrétní nástroje a vlastnosti, které by měl vyvíjený systém obsahovat. První část kapitoly se zaměřuje na popis vybraných nástrojů a jejich využití. Druhá část popisuje vzájemnou provázanost nástrojů a doplňující služby informačního systému.

3.1 Popis navrhovaných nástrojů

Na základě analýzy existujících systémů, typů projektů organizovaných v IAESTE České republiky a konzultací se členy IAESTE byla vybrána sadu nástrojů, které by měly výrazně podpořit práci na projektech. Jedná se o dílčí součásti systému. Jejich vzájemnou provázanost a způsob využití popisují následující kapitoly.

Pro názvy nástrojů jsem použil anglické názvy, protože tato část diplomové práce vznikala během studia v Dánsku a také proto, že systém byl implementován v anglickém jazyce.

3.1.1 Dashboard

Představuje úvodní stranu systému. Obsahuje informace o všech projektech, do kterých je přihlášený uživatel přímo zapojen nebo detail daného projektu. Zobrazuje přehled o aktuální činnosti na projektu a blížících se událostech.

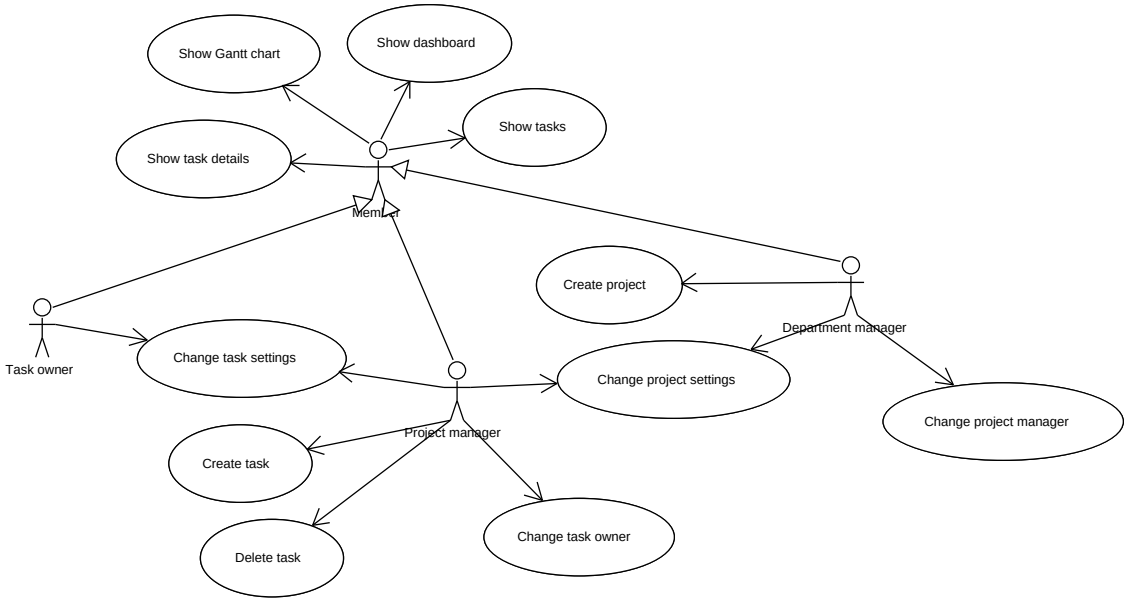
Účelem tohoto nástroje poskytnout uživateli aplikace základní přehled o aktuálním stavu projektů. Nástroj by mohl obsahovat i např. další funkce, jako výpis několika nejnovějších e-mailů v poštovní schránce a podobně.

3.1.2 Project

Project rozděluje systém na jednotlivé projekty, v rámci kterých lze používat dále uvedené nástroje. Use-case diagram je zobrazen na obrázku 3.1.

3.1.3 Task

Task je základní položka v projektu, odpovídající úkolu v Ganttově diagramu nebo v síťovém grafu projektu a byla popsána v kapitole 2.2. Protože je Task (úkol) základní a v našem případě i jedinou strukturovanou složkou systému, je potřeba k němu vztahovat ostatní nástroje, které se tohoto úkolu týkají (nástroje To-Do List, Message, File, Text a Event). Na Task můžeme nahlížet jako na podprojekt, který je členěný pomocí nástroje To-Do List.



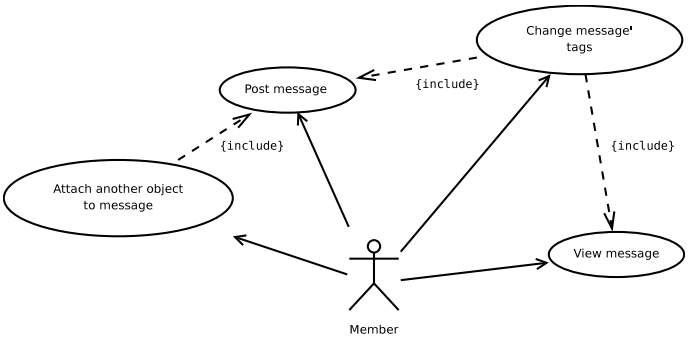
Obrázek 3.1: Use Case diagram projektu a nástroje Task

3.1.4 Message

Message je primárním prostředkem pro komunikaci v týmu. Jedná se o textovou zprávu (obr. 3.2) zařazenou do určité kategorie, případně přiřazenou některému úkolu (Task). Zpráva je viditelná všem členům pracujícím na projektu. Zprávu mohou ostatní uživatelé komentovat pomocí nástroje Comment a tak poskytnout autorovi rychlou zpětnou vazbu.

Použití zpráv je pro informovaní členů týmu o nadcházející události, např. schůzce, o nově vytvořeném souboru nebo report o stavu projektu. Ke zprávě lze přiložit odkaz na Task, To-Do List, Event, File nebo Text.

Při zadávání zprávy lze určit, zda budou (a případně kteří) členové týmu informováni o vytvoření této zprávy e-mailem.



Obrázek 3.2: Use Case diagram nástroje Message

3.1.5 Event

Event je položka v kalendáři. Datum definovaný v kalendáři s názvem a případně popisem. Tento datum může existovat sám (pouze informativní charakter), nebo jej lze přiřadit k Task jako údaj označující začátek, konec Task, nebo událost v rámci Task.

Touto funkcí vznikne společný kalendář projektu i celého centra. Členové tak v jednom kalendáři budou mít přehled o všech událostech, které se týkají projektů do kterých jsou zapojeni.

3.1.6 To-Do List

To-Do List je seznam dílčích úkolů, které už nelze nebo není potřeba v rámci projektu definovat v detailněji. Každý úkol obsahuje informaci o dokončení (dokončen/nedokončen, datum dokončení) a zodpovědného člena týmu (nemusí být). Údaj o dokončení projektu může změnit pouze manager projektu nebo člen přiřazený k tomuto úkolu. To-Do List lze připojit k Task.

Druhým využitím To-Do Listu je týmová spolupráce, *předávání* dílčích úkolů. Příkladem může být například společná příprava internetových stránek, kdy se jako položky To-Do Listu budou zadávat chyby, které je nutné opravit.

3.1.7 Text

Text je nástroj pro společnou tvorbu a úpravu textů. Jedná se o jednoduchý editor, který pomocí velmi jednoduché syntaxe[1] umožní vytvářet texty obsahující tučné písmo, kurzívu, odrážky, nadpisy,... Po uložení se text zobrazí jako klasická HTML stránka. Text musí umožňovat evidenci změn a uzamčení stránky.

Nástroj je navržen pro vytváření zápisů z porad, přípravu textů pro potřeby PR, tvorbu stránek do Průvodce prváka a podobně. Nabízí se srovnání s nástrojem Message, který plní obdobnou funkci. Základním rozdílem je rozdílné určení těchto nástrojů – Message slouží pro jednorázová sdělení a nepředpokládá se evidování změn, Text je určený pro postupné vytváření textu s mnoha změnami.

Výhodou tohoto online editoru je jednoduchá úprava stávajícího textu. Pro opravení chyby nebo doplnění informací není nutné stahovat textový dokument ze serveru, následně editovat a opět nahrávat zpět na server. Tato jednoduchost by měla podpořit uživatele aplikace k větší aktivitě a zároveň zabránit vzniku různých verzí téhož dokumentu, které jsou obtížně slučitelné.

Pro zajištění zpětné vazby budou, obdobně jako u nástroje Message, sloužit komentáře.

Inspirací pro tento nástroj byla služba Writeboard[2] společnosti 37signals, LLC.

3.1.8 File

File je nástroj pro podporu sdílení souborů. Soubor bude možné přes webové rozhraní nahrát do projektu a odkázat na něj z Message nebo Task. Soubor bude možné komentovat a při přepsání se budou ukládat i předchozí verze.

Praktické využití tohoto nástroje je pro sdílení netextových dokumentů, jako jsou obrázky, tabulky, databáze nebo plakáty. Sdílený soubor budou moci uživatelé opět komentovat.

[1] např. wiki nebo texy! (`http://texy.info`) syntaxe

[2] `http://www.writeboard.com`

3.1.9 Comment

Komentář je druhou podstatnou groupware funkcí. Pomocí komentářů Messages, Texts a Files bude snazší získat rychlou zpětnou vazbu na nejrůznější problémy.

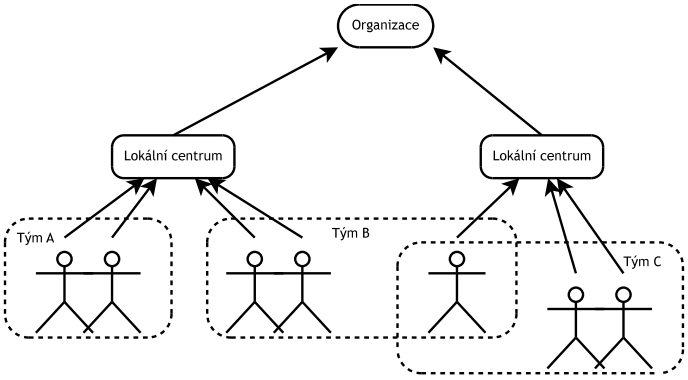
Podobně jako u Message, bude mít autor komentáře možnost určit, zda mají být lidé zapojení v diskusi (nebo i další) informování e-mailem o novém komentáři.

3.2 Hierarchická struktura projektů, nástrojů, organizačních jednotek a uživatelů

Struktura dat uložených v systému je závislá na typu organizace. Protože navrhovaný systém nemá být určený pouze pro použití v jedné organizaci, bylo nezbytné najít kompromis mezi obecností systému a snadností ovládání.

Organizace vedená v systému bude rozdělena na oddělení, departments, která při nasazení v IAESTE ČR budou reprezentovat jednotlivá lokální centra. Tyto oddělení mají v systému svůj soukromý prostor, do kterého ostatní oddělení nemají přístup, pokud to není přímo povoleno vedoucím projektu.

Členové organizace jsou rovněž členěni podle jednotlivých lokálních center – oddělení. V rámci oddělení mohou vznikat týmy, které se mohou skládat ze členů i jiných oddělení. Příkladem takového týmu může být v IAESTE ČR tým zajišťující mezinárodní výměnný program. Vedení týmu je z jednoho lokálního centra, ale členové jsou ze všech lokálních center. Týmy lze využít pro komunikaci a hlavně pro přidělování přístupových práv k projektům. Výslednou strukturu ilustruje obrázek 3.3.



Obrázek 3.3: Navrhovaná struktura týmů a oddělení

Kategorizace projektů a nástrojů je velmi zásadní téma. Vhodná hierarchická struktura může zajistit logické a přesné členění projektů do kategorií. Jedná se o velmi často využívaný přístup, který má ale některé nevýhody. První problém vzniká při ukládání nového projektu nebo nástroje (např. Message) do systému. Uživatel musí vybrat nebo vytvořit odpovídající kategorii, která nejlépe vystihuje daný objekt. Druhá komplikace přichází v okamžiku, kdy se jiný uživatel snaží takto uložený objekt v systému najít.

Tento problém lze dobře ilustrovat na příkladu e-mailového klienta, který umožňuje třídit doručené e-maily do hierarchické struktury složek, kdy je každý e-mail uložen právě v jedné složce. Nabízí se otázka, kam uložíme e-mail, který se týká dvou rozdílných oblastí?

Odpovědí na tento problém bylo v poštovních klientech zavedení technologie takzvaných štítků, nálepek (v angličtině se používají termíny labels nebo tags). E-maily nemusí být tříděny do složek, ale jsou popisovány klíčovými slovy, neboli nálepkami. Jeden e-mail může být popsán více nálepkami zároveň a je dostupný prostřednictvím jakékoli této nálepky. Tento přístup v současnosti využívají téměř všechny nejpoužívanější poštovní programy a je velmi často vyžíván v současných Web 2.0 aplikacích.

Nálepku nebo štítek si můžeme představit jako klasické žluté nalepovací lístky, které mají jasně definované sdělení. Oproti těmto papírovým nálepkám jsme schopni nejen těmito nálepkami objekt popsat, ale rovněž podle těchto nálepek vyhledávat. Předností nálepek je, že uživatele žádným způsobem neomezuje v jejich vytváření a využívání.

3.3 Vzájemná provázanost nástrojů

V předchozí kapitole byly definovány nástroje, které pokrývají různé oblasti týmové spolupráce nebo podpory projektového řízení. Nabízí se otázka, jak vhodně tyto dvě oblasti propojit.

Základní propojení vzniká již při samotném dělení systému na projekty, protože jednotlivé nástroje „existují“ pouze v rámci definovaného projektu. Další propojení je možné buď uživateli „vnutit“, např. omezením vytváření To-Do Listů pouze v závislosti na Task, a tím docílit aktivního využívaní Task a projektového řízení, nebo naopak zvolit opačný přístup, který ponechá konkrétní využívaní nástrojů plně v rukou uživatelů. Protože lze způsob používání systému v budoucnu jen těžko odhadovat a omezování uživatelů nepovažuji za vhodný přístup při tvorbě software, rozhodl jsem se pro druhou možnost. Uživatelé systému budou moci využívat všech nástrojů v systému nezávisle, a bude pouze na nich záležet, jakým způsobem to bude.

Všechny nástroje, kromě komentářů, budou existovat v projektu samostatně. Bude ale možné je vzájemně provázat – odkazovat z jednoho nástroje na druhý.

Obrázek 3.4, který jsem vytvořil při návrhu systému v listopadu 2006, ilustruje, jakým způsobem bude možné jednotlivé nástroje vzájemně provázat.

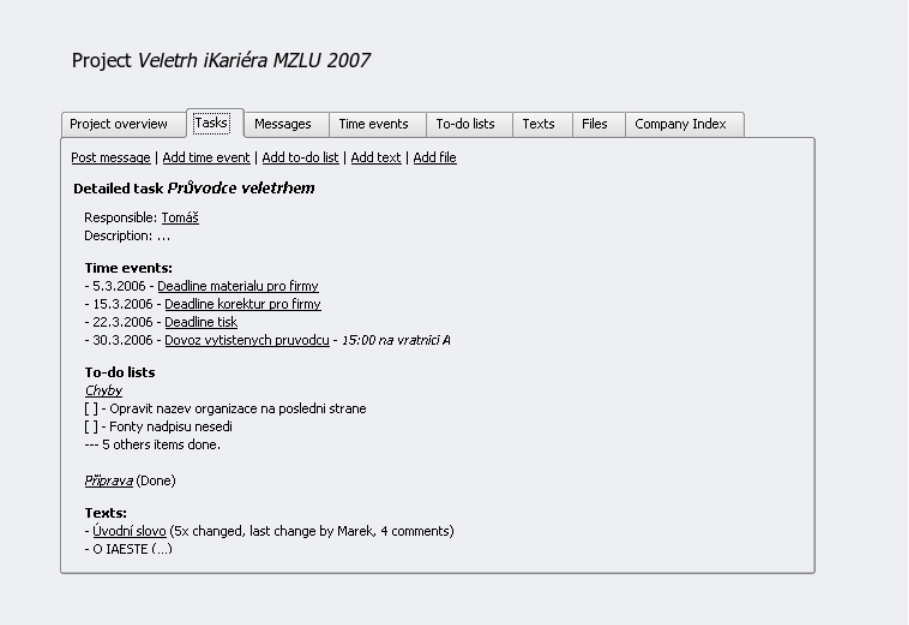
3.4 Přínosy systému

Uložení všech informací o projektu a práci na něm v jednom systému přinese nesporné výhody. Jako hlavní lze považovat:

- *Propojení nástrojů pro projektové řízení a groupware*
  Naplánované úkoly budou viditelné všem členům týmu a lidé, kteří za ně budou zodpovědní je budou moci ihned po dokončení označit jako hotové. Přehled o rozdělení úkolů mezi členy týmu bude mít pozitivní vliv na komunikaci v rámci týmu.

- *Filtrování a řazení podle týmů, lidí, kategorií*
  Rozdělení Messages, Texts, To-do Lists, Files do kategorií a přiřazení k lidem umožní rychlé nalezení hledané informace, případně zobrazení veškeré aktivity (všechny Messages, Comments, To-Do Lists, Tasks atd.) daného člena v rámci projektu nebo všech aktivních projektů.

- *Vyhledávání*
  Protože všechny údaje budou uloženy v jedné databázi, bude možné rychle vyhledávat



Obrázek 3.4: Ukázka provázanosti jednotlivých nástrojů prostřednictím nástroje Task

ve všech oblastech (Tasks, To-Do List, Messages, Texts, Files). Bude tedy možné velmi rychle najít zápis z porady, soubor nebo aktuální stav zpracování úkolu, projektu.

Kapitola 4

Návrh systému

Předchozí kapitola specifikovala všechny požadavky na vlastnosti a funkcionalitu systému. Tato kapitola se zaměří na návrh systému, který bude implementován.

Návrh systému lze rozdělit do několika základních oblastí, které dohromady tvoří celý systém. Jedná se o oblast organizační struktury a uživatelských účtů a oblast nástrojů projektového řízení a týmové spolupráce. Poslední oblastí je návrh zabezpečení systému.

Pro větší názornost a přehlednost diagramů bude popsána každá oblast návrhu projektu samostatně. K popisu návrhu bude využita UML notace. Kompletní diagram tříd je zobrazen na obrázku C.1. Protože jsou kompletní diagramy rozsáhlé a nepřehledné, popisuje tato kapitola pouze nejzajímavější části, které pro názornost zobrazuje na menších diagramech.

4.1 Organizační struktura a uživatelské účty

Požadavky na organizační strukturu a uživatelské účty byly popsány v předchozí kapitole. Doménový model, zachycující statické vazby mezi objekty je zobrazen na obrázku 4.1. Diagram tříd znázorňuje obrázek 4.5 a je popsán v kapitole 4.3.

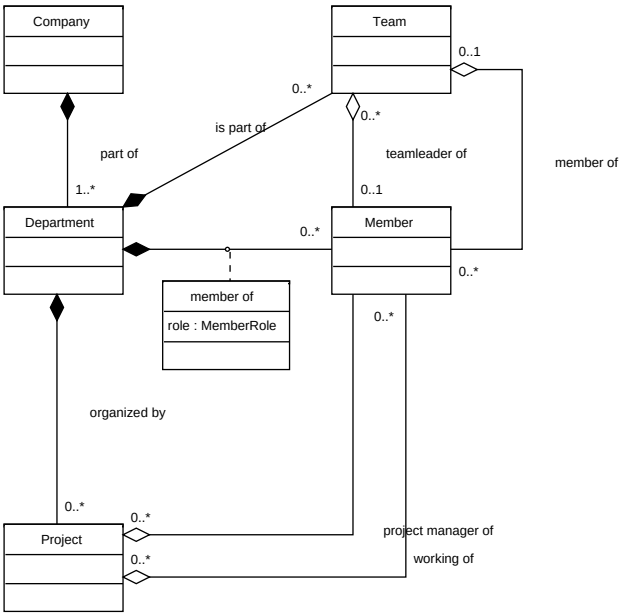
Význam jednotlivých tříd vystihují jejich anglické názvy. Kromě uchování informací poskytují samozřejmě jednotlivé třídy funkční rozhraní. Následující podkapitoly popisují nejdůležitější třídy.

4.1.1 Company

Třída `Company` zastřešuje jednu organizaci, která informační systém používá. Použití jednoho objektu na pomyslném vrcholu hierarchie umožní využívat nainstalovaný software více samostatným organizacím zároveň a s odděleným přístupem (tzn. Software as a Service). Kromě metod poskytují přístup k vlastnostem třídy, poskytuje třída např. metodu `createDepartment`, která slouží k vytvoření nového oddělení organizace. Tato metoda naplňuje návrhový vzor *Creator*, který slouží k vytváření a inicializaci závislých objektů.

4.1.2 Department

`Department` je třída definující oddělení organizace, v případě IAESTE se jedná o lokální centrum. Z požadavků specifikovaných v kapitole 2.3 vyplývá, že není nutné podporovat „zanořování“ oddělení. K rozdělení členů centra do menších skupin slouží týmy – třída `Team`. Diagram 4.1 ukazuje, že třída `Department` musí poskytovat více metod než např. třída `Company`.



Obrázek 4.1: Doménový model organizační struktury informačního systému

Třída `Department` obsahuje množiny objektů, které jsou její součástí. Jedná se o množiny objektů tříd `Team`, `Project`, `Member` a `Tag`, ke kterým třída poskytuje metody podle návrhových vzorů *Creator* a *Expert*.

4.1.3 Member

Třída `Member` je podstatnou třídou v návrhu systému. Instance této třídy reprezentuje jednoho uživatele informačního systému. Obsahuje informace o přihlašovacím jméně uživatele, jeho heslo a členství v lokálním centru (třída `Department`). Dále obsahuje informace o stavu účtu (uzamčený účet, zakázaný).

4.1.4 ProjectGroup

`ProjectGroup` reprezentuje skupinu uživatelů, kteří přímo pracují na projektu. Členství v této skupině má vliv na přístupová práva k projektu (členové této skupiny mají ve výchozím nastavení plný přístup ke všem částem projektu, kromě definování struktury projektu – tuto část může ovlivňovat pouze manažer projektu[1]).

4.2 Nástroje pro podporu týmové spolupráce a projektového řízení

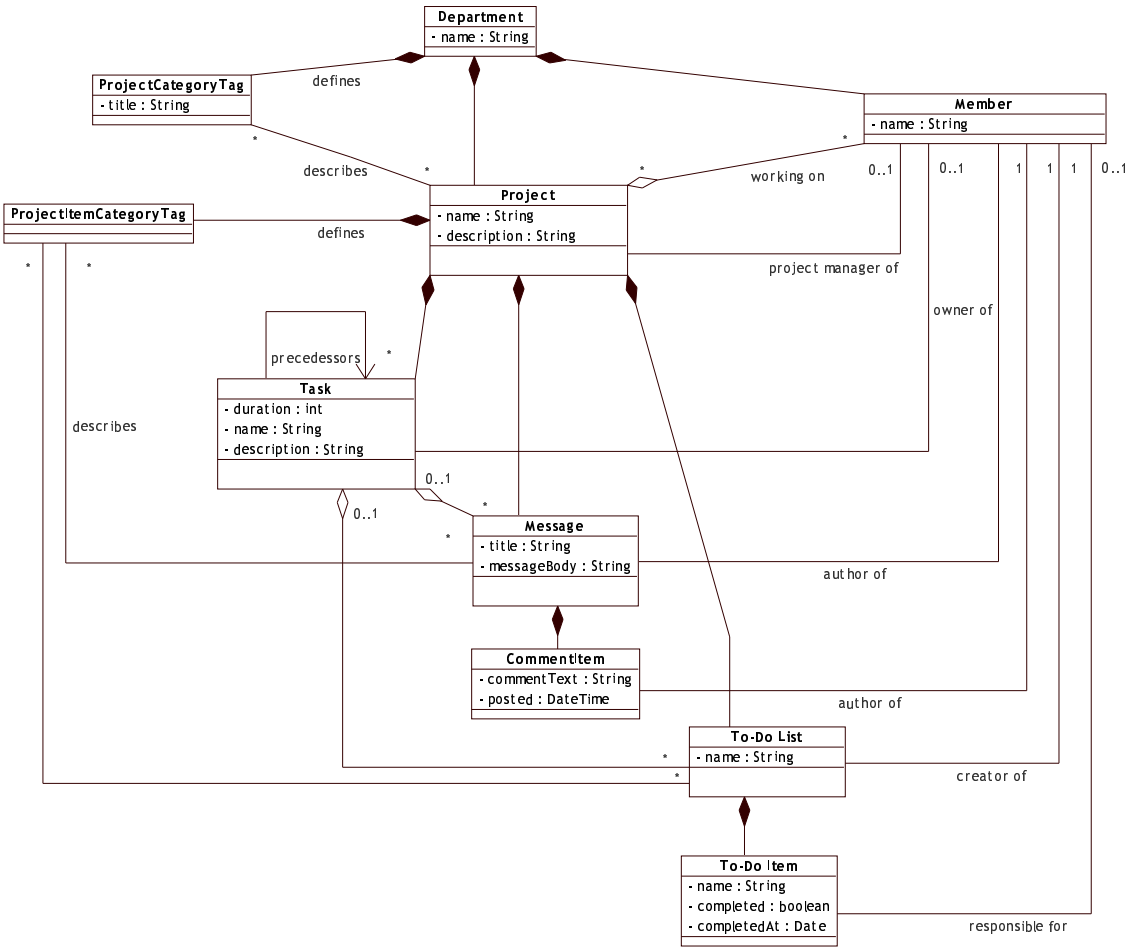
Kvalitní návrh systému zaměřený na modularitu je klíčovým bodem návrhu celé aplikace. Při návrhu jakéhokoli systému nelze nikdy s jistotou určit rozsah systému v budoucnu. Zvláště pak u systému, který slouží pro podporu činnosti uživatelů, kteří pracují na

[1] jedná se o výchozí nastavení pravidel, která se budou pravděpodbně upravovat podle požadavků uživatelů a po zkušenostech s nasazením systému při prvních projektech.

nejrůznějších projektech. Z tohoto důvodu je důležité navrhnout základní strukturu systému dostatečně obecně, aby při požadavcích uživatelů na změnu a zvláště na rozšíření funkcionality systému nebylo nutné upravovat základní prvky systému, ale pouze implementovat novou vlastnost (modul) a propojit ji se stávajícím systémem. Souvisejícím požadavkem je zajištění volného provázání objektů v rámci systému. Tento požadavek byl popsán v kapitole 3.3 a měl by být zaručen nejen pro nástroje, které byly navržené v kapitole 4.2, ale rovněž pro nástroje, které budou implmentovány na žádost uživatelů systému.

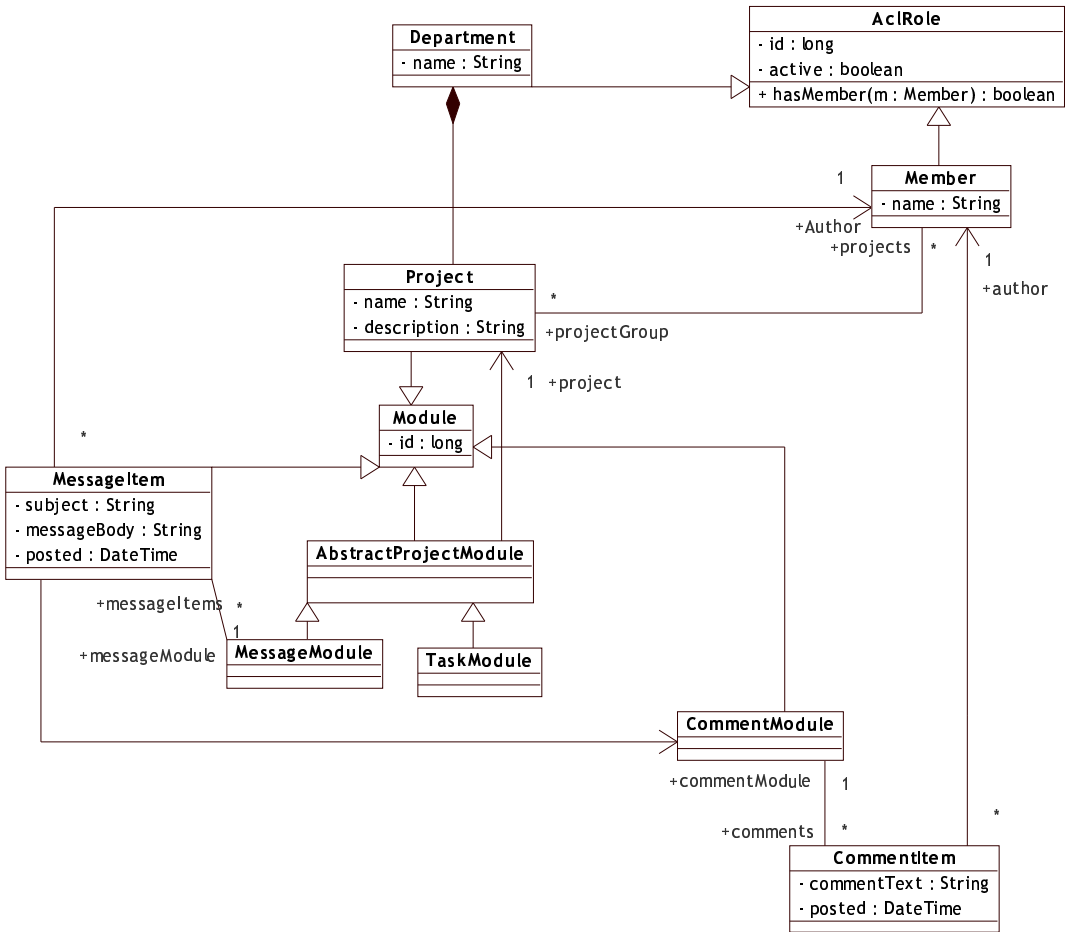
Tato kapitola představí klíčové části návrhu jednotlivých nástrojů. Nejdříve se zaměří na statický model systému znázorňující vazby mezi jednotlivými objekty v systému. Dále bude navržena implementace prostřednictvím diagramu tříd. V poslední podkapitole bude popsán návrh funkční logiky aplikace s využitím návrhových vzorů.

Doménový model na obrázku 4.2 zobrazuje statické vazby mezi prvky systému. Z diagramu je patrné, že centrem všech vztahů je projekt, ke kterému se vážou další objekty. Tento diagram slouží pro ilustraci nejdůležitějších vztahů mezi objekty a byl rovněž použit jako základ pro návrh digramu tříd, který popisuje implementační návrh.



Obrázek 4.2: Doménový model projektu a modulů

Diagram tříd na obrázku 4.3 zobrazuje hierarchii tříd, jejichž předkem je abstraktní třída `Module`. Na diagamu jsou vyznačeny i asociace, avšak z důvodu přehlednosti omezené pouze na část podporující nástroj Message (viz. 3.1.4) a Comment.



Obrázek 4.3: Diagram tříd projektu a modulů

Abstraktní třída `Module` slouží jako předek ke všem třídám, ke kterým potřebujeme definovat v rámci informačního systému přístupová práva – prostřednictvím třídy `AclItem`, dále popsané v kapitole 4.3.3.

4.2.1 AbstractProjectModule

Třída `AbstractProjectModule` je předkem všech tříd, které rozšiřují funkcionalitu třídy `Project`. Definuje atribut `project : Project`, který zajišťuje vazbu nadřazenému projetku. Potomky této třídy jsou nástroje specifikované v kapitole 3.1.

4.2.2 IAttachment

Rozhraní `IAttachment` musí implementovat všechny třídy, pro které potřebujeme zajistit možnost „přiložení“ – vzájemnému provázaní jednotlivých nástrojů. Metody tohoto rozhraní slouží k získání všech nezbytných informací o přiloženém objektu:

- `getAttachmentType() : String` vrací typ přiloženého objektu, který je unikátní pro každou třídu implmentující rozhraní `IAttachment`, např. objekty třídy `MessageItem` vrací vždy `MESSAGE`,
- `getAttachmentTitle() : String` slouží pro získání názvu přílohy,
- `getAttachmentDescription() : String` vrací popisek přílohy. Slouží k zobrazení doplňujících informací,
- `getId() : Long` vrací jedinečný identifikátor přílohy, používá se v kombinaci s metodou `getAttachmentType` pro generování hypertextových odkazů,
- `getIdToShow() : Long` má využití v případě, kdy potřebujeme zobrazit nadřazený prvek. Používá se například u komentářů (`CommentItem`), kdy preferujeme zobrazení komentovaného objektu a všech komentářů, než pouze jednoho komentáře.

4.2.3 ITag

Rozhraní `ITag` je velmi jednoduché – definuje pouze dvě metody, `getId() : Long` a `getTitle() : String`. Toto rozhraní implementují tagy, nálepky používané v informačním systému a cílem tohoto rozhraní je zajistit univerzální přístup k využití tagů.

Rozhraní implementují dvě třídy. `ProjectCategoryTag` slouží pro kategorizaci projektů v rámci lokálního centra. Třída `ProjectItemCategroyTag` se používá pro kategorizaci jednotlivých objektů v rámci projektu. Poznamenejme, že vztah mezi objektem a tagem je z pohledu databázových systémů m:n.

4.2.4 Business Interface

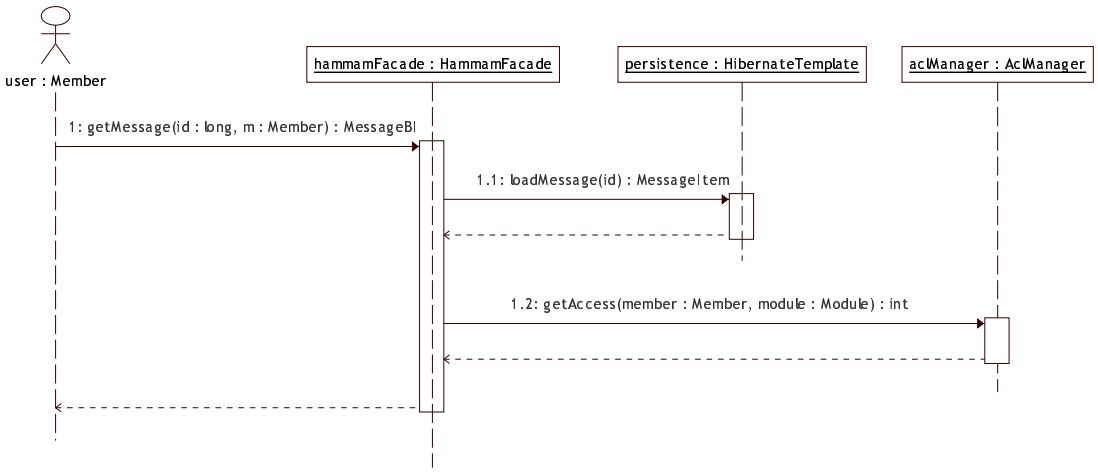
K jednotlivým doménovým objektům, které slouží mimo jiné k mapování dat do perzistenční vrstvy informačního systému, (viz kapitola 5.4) je potřeba definovat jejich aplikační logiku a služby, které mohou poskytovat. Tato logika by měla být odstíněna od implementační technologie, např. od zvolené perzistenční vrstvy. Proto není vhodné zatěžovat doménové objekty aplikační logikou. K tomuto účelu se nejlépe hodí využití rozhraní (interface), které umožňuje zaměnit implementaci rozhraní bez vlivu na ostatní třídy informačního systému. Tento přístup a implementace v diplomové práci je popsán v kapitole 5.3.

Při návrhu byly využity návrhové vzory, které při vhodném využití přispívají k dobrému návrhu systému. Podle [9] se jedná o návrhové vzory *Façade*, *Creator* a *Expert*. Návrhový vzor façade definuje objekt, který je zodpovědný za zpracování požadavků od jednotlivých uživatelů. Tento objekt reprezentuje celý systém a poskytuje přístup ke všem základním doménovým objektům. V navrhovaném informačním systému se jedná o rohraní `HammamFacade`.

HammamFacade

`HammamFacade` poskytuje přístup k jednotlivým klíčovým objektům v systému, jako jsou instance tříd `Company`, `Project`, `Task` nebo `Member`. Tyto objekty jsou různě „zanořené“ pod jinými doménovými objekty, ale potřebujeme k nim přistupovat přímo.
Častým požadavkem je například zobrazení zprávy, instance třídy `MessageItem` definované jedinečným identifikátorem `id`. `HammamFacade` definuje metodu `getMessage(Long id, Member m) : MessageBI`. Parametr `id` udává jedinečný identifikátor zprávy, parametr `m` poskytuje informace o uživateli systému, který se na zprávu dotazuje. Tento parametr

slouží pro ověření přístupových práv k dotazované zprávě. Návaznost jednotlivých kroků zobrazuje diagram na obrázku 4.4. Třída `HibernateTemplate` je část perzistenční vrstvy informačního systému a poskytuje metody pro ukládání a načítání objektů z databáze. Třída `AclManager` poskytuje metody pro získání přístupových práv k jednotlivým doménovým objektům. Tuto třídu popisuje následující kapitola.



Obrázek 4.4: System sequence diagram získání zprávy

Rozhraní `HammamFacade` definuje další metody, jsou to zejména:

- `ProjectBI getProject(Long id, Member m)` pro získání objektu projektu,
- `List<ProjectBI> getAllProjects(Member m)` pro získání všech projektů, ke kterým má uživatel `m` přístup,
- `CompanyBI createCompany(Member m)` pro vytvoření nové společnosti v systému,
- `CompanyBI getCompany(Long id, Member m)` pro získání instance společnosti ze systému,
- `void saveCompany(CompanyBI companyBI, Member m)` pro uložení nastavení společnosti,
- `DepartmentBI getDepartment(Long id, Member m)` pro získání instance oddělení ze systému,
- `MemberBI getMessage(Long id, Member m)` pro získání zprávy,
- ...

CommentFacade

Rozhraní `CommentFacade` poskytuje jednotný přístup ke komentářům, doménovým objektům třídy `CommentItem`. Toto rozhraní využívají *Business Interface* nástrojů, které podporují komentování.

Toto rozhraní je stejně jako `HammamFacade` implementováno v systému podle návrhového vzoru *Singleton*.

Následující podkapitoly popisují tzv. *Business Interface* k doménovým objektům, které toto rozhraní vyžadují. Poznamenejme, že tyto rozhraní již nejsou implementována jako *Singleton*.

ProjectBI

Rozhraní `ProjectBI` „obaluje“ objekty třídy `Project`. Ze všech aplikačních rozhraní je nejobáshlejší, protože definuje většinu metod využívaných v rámci projektu. Rozhraní je definováno z několika důvodů:

1. *bezpečnost* – obdobně jako dříve uvedené rozhraní `HammamFacade` zajišťuje `ProjectBI` kontrolu přístupových práv,

2. *aplikační logika* je přenesená z třídy `Project` do implementace rozhraní. `ProjectBI` definuje několik *Creator* a *Expert* metod a algoritmus pro výpočet kritické cesty projektu. Tyto metody často vyžadují spolupráci s konkrétní implementační technologií. Z tohoto důvodu není vhodné zahrnout aplikační logiku do doménové třídy.

3. *optimalizace* opět souvisí s oddělením aplikační vrstvy od implementace perzistenční vrstvy. Různé implementace perzistenční vrstvy vyžadují jiné způsoby optimalizace. Proto je vhodné definovat rozhraní a ponechat konkrétní implementaci skrytou zbytku aplikační logiky.

MessageBI

`MessageBI` je rozhraní definující aplikační logiku jedné zprávě – nástroji Message. Rohraní definuje metody pro práci s komentářemi a optimalizaci práce perzistenční vrstvy.

4.3 Zabezpečení přístupu k doménovým objektům

Zajistit definování přístupových práv k jednotlivým doménovým objektům informačního systému je bezpochyby jednou z nejdůležitějších částí návrhu. V kapitole 2.3 byly popsány požadavky na přístupová práva a organizační strukturu. Z kapitoly vyplývají dva podstatné požadavky. Prvním je možnost definovat přístupová práva k jednotlivým částem systému nebo projektu s hierarchickou strukturou, kdy objekt níže v hierarchii může mít přístupová práva jiná než objekt v hierarchii výše a rovněž může přístupová práva po nadřazeném prvku dědit. Druhý požadavek spočívá v možnosti definovat přístupová práva nejen ve vztahu k uživatelskému účtu, ale rovněž k týmu, lokálnímu centru nebo celé společnosti.

Existuje mnoho způsobů jak zajistit přístupová práva k jednotlivým částem systému. Za dostatečně univerzální systém lze považovat podle [3] systém pětic (KDO, MODALITA, CO, S ČÍM, KDY), který zajistí dostatečně jemný způsob definování přístupových práv. Jednotlivé prvky pětice vyznačují: který uživatel (KDO), může nebo nemůže (MODALITA) provádět operaci (CO) nad objektem (S ČÍM) v časovém úseku (KDY). Nad tímto systémem lze vystavět další zobecnění definování přístupových práv pomocí skládání podmínek.

Takto navržený systém je ale pro použití v navrhovaném systému zbytečně komplikovaný. Po zvážení všech okolností a modelování různých situací byla vybrána zjednodušená verze výše uvedeného systému. Přístupová práva jsou definována k jednotlivým doménovým objektům pomocí trojice (KDO, S ČÍM, ÚROVEŇ PŘÍSTUPU), kdy KDO představuje uživatele nebo skupinu uživatelů, S ČÍM označuje doménový objekt, např. ToDo List a ÚROVEŇ PŘÍSTUPU určuje vztah uživatele k doménovému objektu. Jednotlivé metody aplikační logiky interpretují ÚROVEŇ PŘÍSTUPU různě podle typu dané metody. V systému jsou definovány čtyři úrovně:

- `ACCESS_NONE` zabraňuje danému uživateli jakýmkoli způsobem přistupovat k danému doménovému objektu,
- `ACCESS_READ` zpřístupňuje uživateli doménový objekt pouze pro čtení,
- `ACCESS_WRITE` dává uživateli možnost daný objekt upravovat. Uživatel ale nemá možnost objekt smazat nebo měnit přístupová práva,
- `ACCESS_OWNER` poskytuje plný přístup k objektu, včetně možnosti objekt smazat a definovat přístupová práva.

4.3.1 Abstraktní třída ACLRole

Diagram tříd organizační struktury informačního systému je zobrazen na obrázku 4.5. Na obrázku nejsou z důvodu přehlednosti vykresleny všechny závislosti mezi objekty. Základem modelu je abstraktní třída `ACLRole`, která slouží jako předek všech tříd, které mají vztah k uživatelským účtům a organizační struktuře. Tato abstraktní třída má využití při definování přístupových práv k doménovým objektům a pro tento účel definuje dvě abstraktní metody, které musí potomci této třídy implementovat:

- metoda `hasMember(Member m) : boolean` vrací hodnotu true, pokud daný objekt obsahuje dotazovaného uživatele.

  V případě implementace této metody v třídě `Member` je prováděn pouze dotaz na identitu dvou uživatelů. Při implementaci ve třídách `Company`, `Department`, `Team` a `ProjectGroup` se ověřuje, zda je dotazovaný uživatel členem instancí těchto tříd. Implementace metody ve třídě `Everyone` vrací vždy hodnotu true.

- Metoda `getType() : Integer` slouží pro řazení priorit přístupových práv. Využití vychází z předpokladu, že přístupová práva definovaná pro konkrétního uživatele mají větší prioritu než práva definovaná pro celé lokální centrum (třída `Department`).

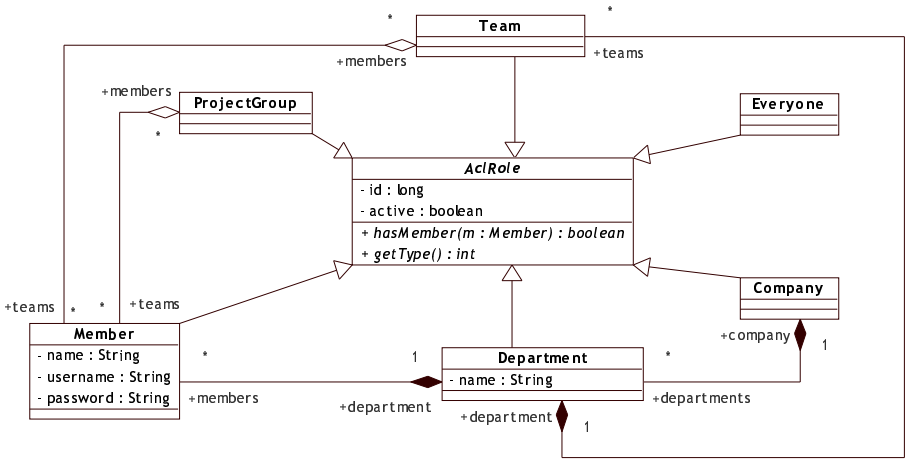
  Priority jsou definovány v tomto pořadí, sestupně od nejvyšší priority:
  `Member`,
  `ProjectGroup`,
  `Team`,
  `Department`,
  `Company`,
  `Everyone`.

Význam většiny tříd je zřejmý. Za povšimnutí stojí třída `Everyone`, která slouží ke zrušení dědičnosti přístupových práv k objektu. Její využití je popsáno v následující kapitole.

4.3.2 Hierarchie přístupových práv

V předchozím textu jsme se několikrát setkali s pojmem hierarchie přístupových práv, ale zatím nebyl tento pojem přesně popsán a vysvětlen.

Trojice (KDO, CO, ÚROVEŇ PŘÍSTUPU) umožňuje definovat libovolná přístupová práva ke kterémukoli doménovému objektu v systému. Tento systém je pro nás dostatečně jemný a přesný, ale má jednu zásadní nevýhodu. Tou je potřeba definovat přístupová práva pro každý doménový objekt zvlášť a tyto přístupová práva budou velmi pravděpodobně pro

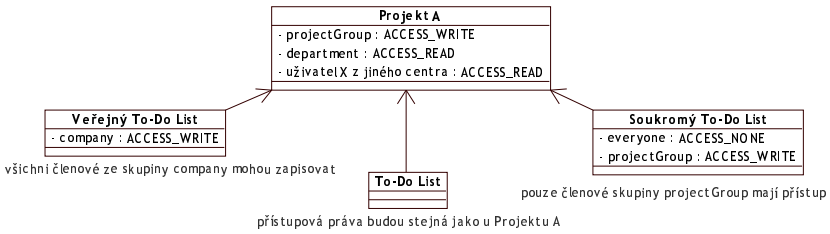


Obrázek 4.5: Class diagram tříd organizační struktury

většinu objektů v rámci projektu stejná. Problém opakovaného generování přístupových práv snadno odbourá použití návrhového vzoru *Factory*[9]. Tento návrhový vzor nám umožní vytvářet objekty s předdefinovanými vlastnostmi a vazbami. Problém nastává při změně přístupových práv na vyšší úrovni, např. na úrovni celého projektu. S takovou změnou je nutné přepsat nebo doplnit všechny trojice (KDO, CO, ÚROVEŇ PŘÍSTUPU) u všech objektů v projektu.

Elegantním řešením je využít hierarchickou strukturu objektů v rámci projektu a dědičnost. Doplněním základní abstraktní třídy `Module` o vlastnost `parent` typu `Module`. V případě prvku na vrcholu hierarchie nabývá tato vlastnost hodnoty `Null`, v opačném případě odkazuje na nadřazený prvek, ze kterého se dědí všechna přístupová práva. Tyto přístupová práva může objekt přepsat vlastní konfigurací, která má samozřejmě větší prioritu než práva definovaná v nadřazeném objektu.

Jednoduchý příklad využití znázorňuje obrázek 4.6[2], ve kterém je `Veřejný To-Do List` použit pro reportování chyb od uživatelů (kohokoli z celé organizace) a naopak `Soukromý To-Do List` je skryt před všemi uživateli kromě projektového týmu.



Obrázek 4.6: Ukázka využití hierarchie přístupových práv (popis syntaxe v textu)

[2]Nejedná se o diagram podle UML notace. Cílem obrázku je naznačit hierarchii objektů v systému – šipky znázorňují hierarchii, obdélníky instance jednotlivých nástrojů vytvořených v systému.

4.3.3 Třída AclManager

Vyhodnocování přístupových práv je rozloženo mezi tzv. *business rozhraní* popsaná dříve a třídu `AclManager`, která je jádrem vyhodnocování přístupových práv.

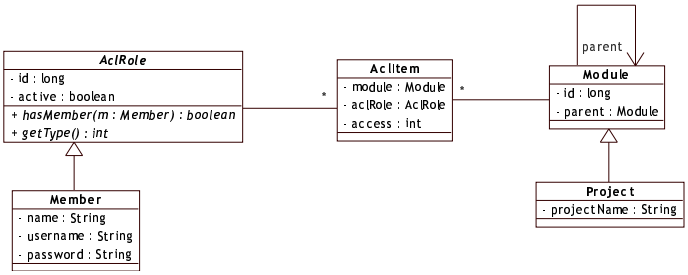
Metoda využívaná ostatními částmi systému se jmenuje `getAccessMode(Member member, Module module) : int`. Metoda vrací nejsilnější přístupové právo (tzn. to, kterým `member` získá největší pravomoce) k danému objektu `module`. Vyhledávání je zajištěno rekurzivním algoritmem:

```
if (module == null)
  return ACLItem.ACCESS_NONE;

List<ACLItem> moduleACLs = getModuleACLs(module);

Iterator<ACLItem> it = moduleACLs.iterator();
while (it.hasNext()) {
  ACLItem aclItem = it.next();
  if (aclItem.getAclRole().hasMember(member))
    return aclItem.getAccess();
}

return getAccessMode(member, module.getParent());
```

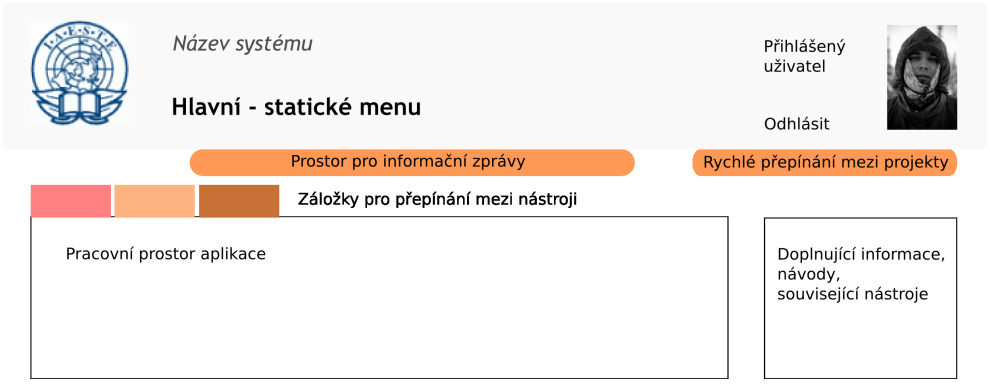


Obrázek 4.7: Třída AclItem pro zajišťující definici přístupových práv

4.4 Grafické uživatelské rozhraní

Návrh uživatelského rozhraní vycházel ze zkušeností s používáním obdobných systémů, analýzy populárních internetových aplikací – nejen aplikací pro správu projektů nebo plánování času, ale i méně „pracovních“ nástrojů, např. služeb GMail, Flickr, del.icio.us, last.fm nebo MySpace. Tedy služeb, které aktivně využívají milióny uživatelů. Jednou ze společných vlastností těchto služeb bylo jednoduché uživatelské rozhraní a pouze omezené používání grafických prvků.

Obrázek 4.8 představuje návrh rozložení základních prvků. Stránka je navržena s pevným rozdělením jednotlivých oblastí, které se při práci s aplikací nepřesouvají. Horní část je statická, obsahuje informace o přihlášeném uživateli, název systému a menu pro přepínání mezi hlavními částmi systému. Spodní část strány se mění prakticky s každým požadavkem uživatele, rozložení ale zůstává stále stejné. Prostor na pravé straně souží k zobrazení doplňujících informací k zobrazeným datům a službám, které může uživatel využít. Snímek obrazovky implementovaného systému je na obrázku 5.3.



Obrázek 4.8: Návrh rozložení uživateského rozhraní aplikace

Kapitola 5

Implementace

Předchozí kapitola představila návrh budoucího systému. Navazujícím krokem při zpracovávání projektu je zvolení vhodné implementační technologie. Přestože už v začátku projektu padlo rozhodnutí implementovat systém v programovacím jazyku Java[1], následný výběr vhodných technologií pro implementaci jednotlivých vrstev aplikace už tak přímý nebyl. Kapitola 5.1 popisuje volbu a hodnocení jednotlivých technologií. Následující kapitoly popisují implementaci jednotlivých částí systému.

5.1 Výběr a popis technologií

Základní omezení, které ovlivnily volbu technologií byly popsány v kapitole 2.6. Směrodatným prvkem je použití internetového prohlížeče jako klientské technologie. Volba, která ovlivnila celý další vývoj projektu spočívala ve výběru programovacího jazyka a platformy. Pro výběr platformy byly rozhodující tyto oblasti:

rozšířenost – systém bude v budoucnu spravovat nejen autor, ale i další administrátoři a programátoři. Proto je rozšířenost a obecná znalost použité platformy velmi důležitá,

cena je důležitým hlediskem pro nasazení v neziskových organizacích. Cenu je potřeba hodnotit z více hledisek: cena za vývoj (platy programátorů s danou specializací), cena za vývojové prostředí a cena za provoz aplikace (pronájem, hostování serveru s danou technologí nebo nákup licence aplikačního serveru),

stabilita a zpětná kompatibilita jako kritérium pro dlouhodobý provoz aplikace,

efektivita vývoje , na které přímo závisí cena vývoje. Efektivita vývoje je závislá na dostupnosti kvalitního vývojového prostředí (IDE), ladících a testovacích nástrojů a také rychlosti a složitosti vývoje daného druhu aplikace a zvolené platformě,

rozšiřitelnost o případné nové technologie, které nejsou součástí prvotního zadání projektu.

Po zvážení všech těchto kritérií, probíhal výběr mezi platformami Microsoft .NET 2.0[2], Java Enterprise Edition a PHP 5.2[3]. Ostatní platformy, programovací jazyky ne-

[1] `http://java.sun.com`
[2] `http://msdn.microsoft.com/netframework`
[3] `http://www.php.net`

byly do výběru zařazeny hlavně z hlediska rozšířenosti, přestože nabízely zajímavé vlastnosti a měli jisté výhody, např. vysoká efektivita vývoje prostřednictvím webového open-source frameworku Ruby on Rails™[4] nebo velmi přehledná organizace kódu (a tím pádem snadná udržovatelnost aplikace) a množství dostupných knihoven pro programovací jazyk Python™[5].

Microsoft .NET Framework je v současné době velmi populární a rozšířenou technologií nejen v komerční sféře. Umožňuje vytvářet aplikace v různých programovacích jazycích včetně velmi dobře navrženého jazyku `C#`. Obsahuje kvalitní, komponentově orientovaný webový framework, propojení k databázím a rovněž je k dispozici vývojové prostředí MS Visual Studio dostupné v základní variantě zdarma.

Největší výhodou a naopak i hedikapem platformy Microsoft .NET je těsná závislost na produktech společnosti Microsoft. Podpora silné společnosti zaručuje další rozvoj této technologie. Nevýhodou je nutnost zakoupení licence na Microsoft Windows Server. Existuje sice open-source implementace .NET Frameworku nazvaná Mono[6], která umožňuje spouštět klientské i serverové aplikace využívající .NET Framework pod jinými operačními systémy než Microsoft Windows, ale dosud není plně kompatibilní s .NET Frameworkem verze 2.0, který obsahuje podstatné vlastnosti pro vývoj internetových aplikací.

Programovací jazyk PHP je nejrozšířenějším skriptovacím jazykem používaným na internetových serverech. Mezi největší výhody patří velmi snadný a rychlý vývoj jednoduchých internetových aplikací, velká nabídka hostování aplikací, ať už komerční nebo zdarma. Pro vývoj složitějších aplikací ale PHP není vhodná volba, zvláště z důvodu udržovatelnosti kódu, změnami chování některých funkcí při přechodu na novou verzi, které často způsobují problémy při provozování starších aplikací na nových verzích PHP. Pro vývoj většího projektu je nedostatečná podpora objektově orientovaného programování.

Java Enterprise Edition je serverová platforma programovacího jazyka *Java™*. V současné době zahrnuje desítky nejrůznějších standardů a technologií od mnoha výrobců. Mezi výhody JEE patří kvalitní, platformě nezávislý, objektově orientovaný programovací jazyk Java, běhová prostředí (Java Virtual Machine) od různých výrobců (Sun Microsystems, IBM, BEA Systems) dostupná pro mnoho operačních systémů a pod různými licencemi. Přestože je vývoj aplikací v programovacím jazyce Java a `C#` velmi podobný (jazyk `C#` je silně inspirovaný Javou), lze na JEE v mnoha ohledech nahlížet jako na opak Microsoft .NET Frameworku. Zatímco .NET Framework je prakticky celý postaven na technologiích jednoho výrobce, pro JEE existuje velké množství implementací obdobné funkcionality. To sebou přináší nespočet výhod, ale rovněž náročnost výběru vhodné technologie.

Jednou z klíčových nevýhod JEE platformy je problém s hostováním projektů. Zatímco PHP projekty lze hostovat velmi snadno a to i zdarma, pro JEE projekty je hosting webových aplikací podstatně dražší.

Po zvážení všech vlastností jednotlivých technologií byla vybrána platforma Jave Enterprise Edition. Následující kapitola popisuje jednotlivé prvky vybrané platformy, které byly využity při vývoji aplikace.

[4] `http://www.rubyonrails.org`
[5] `http://www.python.org`
[6] `http://www.mono-project.com`

5.1.1 Aplikační rámec

Přestože lze informační systém naprogramovat pouze s využitím technologie JSP, je vhodnější oddělit jednotlivé vrstvy aplikace a to nejlépe prostřednictvím tzv. aplikačního rámce (aplikačního frameworku), který nám poskytne služby řídící běh celé aplikace. Ve velkých systémech se pro tyto účely používá robustní JEE standard EJB (Enterprise JavaBeans), který poskytuje podporu pro tvorbu modulárních distribuovaných aplikací. Tento standard je ale pro malé projekty příliš těžkopádný a náročný a rovněž vyžaduje spuštění EJB v prostředí aplikačního serveru.

Proto vznikly frameworky, které poskytují obdobnou funkcionalitu jako EJB, ale za mnohem jednodušších podmínek pro programátora a menších nároků na hostující server. Těmto frameworkům většinou stačí pro běh aplikace pouze servrletový container, např. Apache Tomcat nebo Jetty. V současnosti nejpopulárnějším a nejrozšířenějším frameworkem toho typu je open source *Spring framework*[7] vyvýjený od roku 2003 společností Interface21. Není bez zajímavosti, že popularita Spring Frameworku a jeho klíčových vlastností se promítla do návrhu nového standardu EJB 3.0 a výrazně jej ovlivnila. Z další frameworků můžeme jmenovat např. PicoContainer[8] nebo google-guice[9].

Protože využití Spring Frameworku výrazně ovlvnilo celý proces implementace aplikace, představí následující odstavce nejdůležitější vlastnosti a části frameworku. Použití jednotlivých komponent v celkové architektuře implementová aplikace názorně zobrazuje obrázek 5.1.

Inversion of Control je návrhový vzor, který představuje možnost přesunout odpovědnost za vytváření a běh jednotlivých objektů z aplikace do aplikačního rámce. Veškeré činnosti spojené s nahráváním objektů, připojením k databázi nebo konfigurace aplikace obstarává aplikační rámec. Samotná aplikace pouze čeká na volání jednotlivých metod.
Spring framework dále implementuje úžeji specifikovanou verzi návrhového vzoru Inversion of Control – Dependency Injection. Ta již podle názvu spočívá ve *vnucení* závislostí jednotlivým objektům, a to aplikačním rámcem. Spolupracující objekty nejsou provázány v kódu aplikace voláním metod, ale prostřednictvím metadat v konfiguračním XML souboru (lze použít i konfiguraci pomocí anotací) s využitím `set` a `get` metod POJO objektů[10].

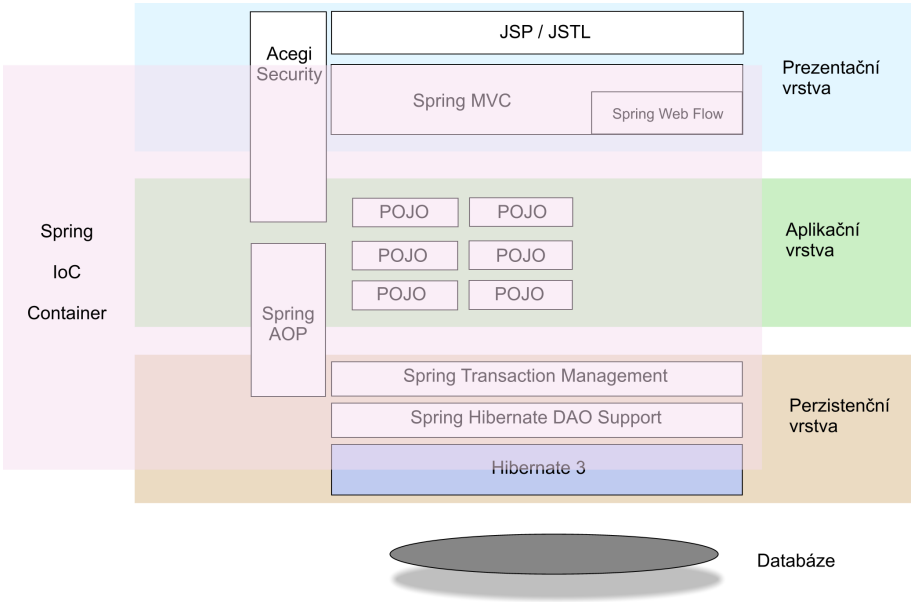
Tento, na první pohled zbytečný a komplikující přístup, přináší spoustu výhod, které se projevují zvláště při správě aplikace a dlouhodobém vývoji, protože celá konfigurace aplikace je popsána jednotným způsobem v jednom místě. Tato konfigurace navíc nevyžaduje implementaci speciálních rozhraní pro komunikaci mezi objekty a jejich konfiguraci.

Představme si, že v naší aplikaci máme třídu AclManager, která byla popsána v předchozí kapitole. V každém business rozhraní využíváme služby této třídy a protože se jedná o třídu podle návrhového vzoru Singleton, získáváme ji při použití statickou metodou `AclManager.getInstance()`. Pokud v budoucnu budeme potřebovat nahradit tuto třídu třídou novou, budeme muset ve všech třídách, které třídy `AclManager` využívají upravit odpovídající kód.

[7] `http://springframework.org`
[8] `http://www.picocontainer.org`
[9] `http://code.google.com/p/google-guice`
[10] POJO je zkratka z anglického spojení Plain Old Java Object a dá se volně přeložit jako starý dobrý Java objekt. Toto označení není negativní a má vyjadřovat jednoduchost oproti komplikovaným Enterprise JavaBeans objektům. POJO objekt je takový Java objekt, který má bezparametrický konstruktor a privátní vlastnosti třídy, ke kterým lze přistupovat prostřednictvím veřejných `get*` a `set*` metod.

Pokud budeme *programovat rozhraním*[11] a použijeme IoC controller, bude se tato změna týkat pouze jednoho konfiguračního souboru.

Na obrázku 5.1, znázorňujícím využití jednotlivých komponent v impolementovaném informačním systému, můžeme vidět, že služby Spring IoC kontejneru využívají všechny vrstvy aplikace.



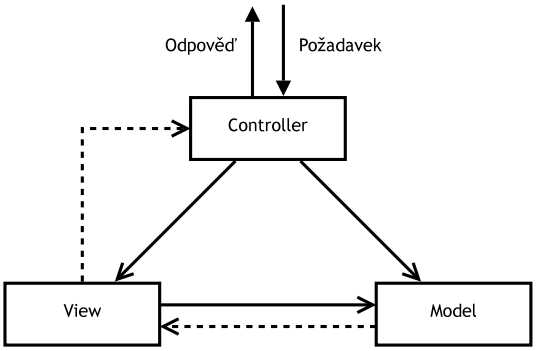
Obrázek 5.1: Architektura implmentovaného informačního systému

Web MVC Framework je jednou z mála oblastí, kterou Spring Framework přímo implentuje – v ostatních oblastech Spring „pouze“ podporuje snadné propojení existujících frameworků, případně přidává k těmto frameworkům doplňující funkcionalitu (např. univerzální hierarchii výjimek pro přístup k perzistenční vrstvě). Web MVC Framework poskytuje implementaci návrhového vzoru Model-View-Controller, který je doporučeným přístupem pro návrh aplikací využívajících uživatelské rozhraní (tedy nejen internetových aplikací).

Model-View-Controller zajišťuje oddělení aplikační logiky (modelu) a prezentační vrstvy (view) aplikace zavedením třetí části, tzv. controlleru. Diagram Model-View-Controlleru je zobrazen na obrázku 5.2, plné čáry označují přímou komunikaci (volání metod), tečkované čáry data a události.

Spring Web MVC Framework poskytuje přímou podporu pro technologie generující HTML kód: JSP, Velocity, Tiles, iText a POI.

[11] Programování rozhraním je jednou z best practices[6], která je silně využívaná a podoporována Spring Frameworkem. Spočívá v použití rozhraní v maximální možné míře, zvláště pak u vlastností objektů nastavovaných prostřednictvím IoC. Tento přístup umožňuje maximální flexibilitu při změně implementace tříd.



Obrázek 5.2: Model-View-Controller

5.1.2 Prezentační vrstva

Uživatelé přistupují k informačnímu systému prostřednictvím internetového prohlížeče – využívají protokol HTTP a pro zobrazení dat HTML stránky. Pro platformu Java existuje velké množství nástrojů, které umožňují generování HTML stránek. Mezi nejpoužívanější patří technologie JSP/JSTL, JavaServer Faces, Jakarta Strutcs, Webwork nebo Tapestry. Pro implementaci systému byla zvolena technologie JSP (JavaServer Pages) s využitím JSTL (Java Standard Tag Library) a doprogramováním vlastních JSP tagů. Spring MVC velmi dobře spolupracuje s technologií JSP a poskytuje dodatečné tagy, které usnadňují obsluhu formulářů, podporu témat vzhledu a tvorbu vícejazyčných aplikací.

Nicméně, díky využití Spring Frameworku lze snadno v budoucnu nahradit tuto technologii jinou, poskyující např. snadnou implementaci AJAX[12] technologie a komponentově orientovaný přístup.

JavaServer Pages v kombinaci s Java Standard Tag Library poskytuje velmi silný nástroj pro tvorbu dynamického obsahu internetových stránek. JSP umožňuje vkládat do stránky klasický Java kód, tzv. scriptlet, ale tento přístup není doporučován. Vhodnější je implementovat vlastní JSP tagy, které odstíní konkrétní implementaci a nesvádějí programátora k vytváření aplikační logiky uvnitř JSP stránek. Od verze 2.0 je součástí JSP i Expression Language (EL) poskytující jednodušší přístup k aplikačním datům (typicky předanými prostřednictvím MVC Frameworku).

5.1.3 Perzistentní vrstva

Nejrozšířenějším a nejefektivnějším způsobem perzistence dat v informačních systémech je využití relačních databázových systémů. Platforma Java poskytuje standardní rozhraní, které může být využito pro přístup k databázi – JDBC™(Java Database Connectivity). Knihovny nezbytné pro přístup ke konkrétní databázi poskytuje prakticky každý výrobce databázových systémů.

JDBC je standardem, který využívají pokročilé nástroje pro přístup k databázi. Tyto nástroje poskytují různou funkcionalitu, ať už se jedná o zajištění transakcí, database connections pooling[13], abstrakci typu databáze nebo objektově relační mapování.

[12] AJAX (*Asynchronous JavaScript and XML*) je označení technologie pro vývoj interaktvních stránek, které pro aktualizaci dat na stránce nevyžadují znovunačtení celé stránky, ale využívají asynchronních dotazů na server a na základě dat získaných ve formátu XML mění obsah stránky.

[13] *database connections pooling* – technologie umožňující zefektivnit připojování aplikace k databázi.

Posledně zmíněná oblast se stává v poslední době velmi populární a svědčí o tom i vytvoření standardu JPA (Java Persistence API), který je součástí nového standardu pro vývoj enterprise aplikací EJB 3.0. Objektově relační mapování je způsob, jak ukládat objekty a vazby mezi objekty do relační databáze.

V současné době existuje několik frameworků pro platformu Java, které poskytují objektově relační mapování. Mezi nejrozšířenější patří open-source Hibernate[14], za kterým stojí společnost Red Hat Middleware, Oracle TopLink[15] nebo v porovnání s předchozími jednodušší framework iBATIS Data Mapper[16]. Uvedené nástroje se liší v rozsahu poskytované funkcionality (např. způsoby mapování hierarchie objektů, vztahů mezi objekty), rozsahem podporovaných databází, úrovní abstrakce relační databáze nebo způsobem popisu mapování objektů do databáze.

Objektově relační mapování přináší do vývoje informačních systémů spoustu pozitivních prvků, ale zároveň několik negativních. Mezi největší výhody patří zejména vyšší produktivita při vývoji aplikace – většinu běžných problémů lze mapovat bez využití jazyka SQL a nahrání objektu nebo uložení objektu do databáze je otázkou jednoho řádku kódu. Vyšší produktivity lze ale docílit pouze dobrou znalostí této technologie, která v sobě skrývá nemalé množství komplikovaných částí. Za další výhodu lze považovat odstínění konkrétní databáze od návrhu mapování. Každý databázový systém se od ostaních liší v různých více či méně podstatných detailech. Ať už se jedná o rozšíření syntaxe SQL jazyku, podporované datové typy nebo způsoby definování přísupových práv. Objektově relační framework tyto rozdíly skrývá za svou implementací, což zajišťuje snadnou přenositelnost mezi různými databázovými systémy. Nevýhodou této vlastnosti je, že objektově relačním nástrojem vygenerované schéma databáze není optimalizované pro aplikací využívaný databázový server[17]. Další velmi zajímavým přínosem je tzv. *lazy loading*, což je metoda, kterou se optimalizuje přístup k databázi – data, např. kolekce jsou dotazovány z databáze, až jsou doopravdy potřeba. Od tohoto dodatečného dotazování je programátor plně odstíněn. Způsob využívání lazy loadingu lze samozřejmě různým způsobem optimalizovat pro jednotlivé třídy.

Hibernate je v současnosti jedním z nejpopulárnějších ORM nástrojů. Díky velkému spektru poskytovaných funkcí, včetně polymorfického mapování, se ukázal dobrou volbou pro implementaci informačního systému. Nutno poznamenat, že přestože je na první pohled programování s Hibernate velmi snadné, přináší používání tohoto nástroje spoustu úskalí i komplikací. Výhodou Hibernate je i velmi dobrá integrace se Spring Frameworkem.

5.2 Implementace prezentační vrstvy

Pro implementaci prezentační vrsty byla vybrána kombinace Spring MVC frameworku pro zajištění obsluhy formulářů a HTML stránek a JSP/JSTL pro vytváření výstupního HTML kódu.

V kódu JSP stránek bylo využito několik vlastních JSP tagů. Jedná se o tagy zajišťující

Knihovna poskytující funkci database pooling sama na základě nastavení a běhu aplikace určuje, kdy se mají jednotlivá připojení k databázi uzavírat nebo otevírat. Tímto způsobem se optimalizuje přístup k databázi, protože navázání nového připojení k databázi je časově velmi náročné.

[14] `http://www.hibernate.org`

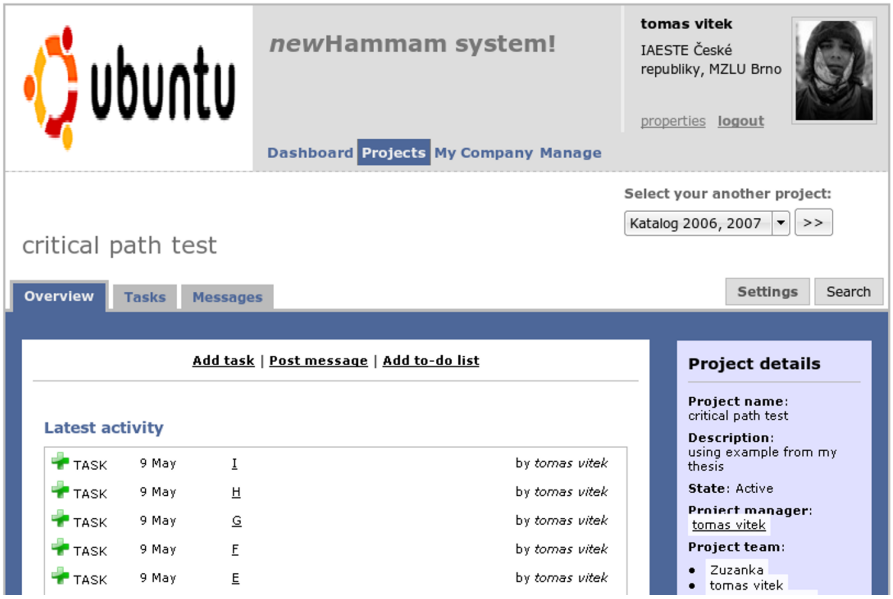
[15] `http://www.oracle.com/technology/products/ias/toplink`

[16] `http://ibatis.apache.org`

[17] Hibrnate umožňuje ovlivnit, jakým způsobem se bude databázové schéma generovat nebo lze přenechat vytvoření schématu na programátorovi.

vykreslení Ganttova diagramu, obsluhu zaškrtávacích políček ve formulářích (tuto oblast nástroje Spring MVC dostatečně nepokrývají), správné formátování textu zadaného pomocí nástroje Message nebo Comment a tag pro překlad adresy přílohy (Attachment).

Vlastní HTML stránka je rozčleněna do několika samostatných logický oblastí, které jsou následně vizuálně upraveny do výsledného vzhledu pomocí kaskádovacích stylů CSS. Použití CSS pro formátování vzhledu dokumentů je silně doporučovanou praktikou, protože neovlivňuje strukturu dokumentu a rovněž umožňuje definovat různé vzhledy stránky pro různá výstupní zařízení (obrazovka, tiskárna, čtecí zařízení…). Obrázek 5.3 zachycuje implementovaný informační systém.



Obrázek 5.3: Snímek obrazovky implementovaného systému

5.3 Implementace aplikační vrstvy

Využití Spring Frameworku mělo minimální vliv na návrh aplikační vrstvy aplikace. Díky využití návrhového vzoru Dependency Injection nebylo třeba psát speciální rozhraní k vlastním třídám. *Programování rozhraním* umožnilo odstínit funkční rozhraní business objektů od konkrétní implementace.

Příklad představí implementaci jednoho business rozhraní – `ProjectBI`. Rozhraní obaluje doménový objekt `Project` a přidává k němu vlastní funkcionalitu, která je specifická pro implementaci systému a zvolenou perzistenční vrstvu. Následující kód ukazuje definici rozhraní, které využívají ostatní části aplikace, např. MVC controller.

```
public interface ProjectBI {
  Project getProject();
  void setProject(Project project);
  MessageBI createMessage(Member m);
  MessageBI getMessage(Long ID, Member m);
  long getMessagesCount();
  ...
}
```

Zkrácený výpis třídy `ProjectBIHibernate` ukazuje využití Dependency Injection v praxi. Třída obsahuje pouze definici vlastností třídy a `set*` metody pro počáteční nastavení těchto vlastností Spring Frameworkem. Za povšimnutí stojí, že všechny vlastnosti, kromě třídy `Project`, jsou opět rozhraní.

Na řádku 15 využívá metoda `getMessage()` služby Spring Frameworku pro získání instance rozhraní `MessageBI` z aplikačního kontextu pod jménem `messageBI`. Metoda `getBean` vrátí plně připravený objekt včetně všech závislostí. Definice tohoto rozhraní je definována obdobně jako níže uvedená konfigurace `projectBI`.

```
1  public class ProjectBIHibernate extends HibernateDaoSupport implements
2      ProjectBI, ApplicationContextAware {

3    private Project project;
4    private HammamFacade hammamFacade;
5    private ACLManager aclManager;
6    private ApplicationContext applicationContext;

7    void setProject(Project project) {
8      this.project = project;
9    }
10    ...

11    public MessageBI getMessage(Long ID, Member m) {
12      MessageItem message = (MessageItem) getHibernateTemplate().load(
13        MessageItem.class, ID);

14      if (aclManager.getAccessMode(m, message) != ACLItem.ACCESS_NONE) {
15        MessageBI mbi = (MessageBI) applicationContext.getBean('messageBI',
16          MessageBI.class);
17
18        mbi.setMessageItem(message);

19        return mbi;
20      } else
21        throw new AccessDeniedException('User ' + m.getName()
22          + ' cannot read message with id ' + ID.toString() + '.');
23    }
24    ...
25  }
```

Následující ukázka popisuje definici `projectBI` v definičním souboru Spring Frameworku. Třída `ProjectBIHibernate` (řádek 8) je obalena třídou `TransactionProxyFactoryBean` (řádek 2), která zajišťuje deklarativní podporu transakcí.
Na řádku 9 je zakázáno použití návrhového vzoru singleton pro vytváření definovaného objektu. Tím zajistíme, že při každém dotazu na tento objekt (jak je ukázáno na řádku 15 předchozího výpisu kódu), bude vytvořena nová instance této třídy. Naopak u rozhraní `AclManager` z principu vyžadujeme použití návrhového vzoru singleton.

```
1  <bean id='projectBI'
2    class='org.springframework.transaction.interceptor.TransactionProxyFactoryBean'
3    singleton='false'>
4    <property name='transactionManager' ref='transactionManager' />
5    <property name='target'>
6
7      <bean
8        class='com.blbeckove.viitan.hammam.business.hibernate.ProjectBIHibernate'
9        singleton='false'>
10       <property name='sessionFactory'
11         ref='mySessionFactory'>
12       </property>
13       <property name='aclManager' ref='aclManager'></property>
14       <property name='hammamFacade' ref='hammamFacade'></property>
```

```
15      </bean>
16
17    </property>
18    <property name='transactionAttributes'>
19      <props>
20        <prop key='*'>PROPAGATION_REQUIRED</prop>
21      </props>
22    </property>
23  </bean>
```

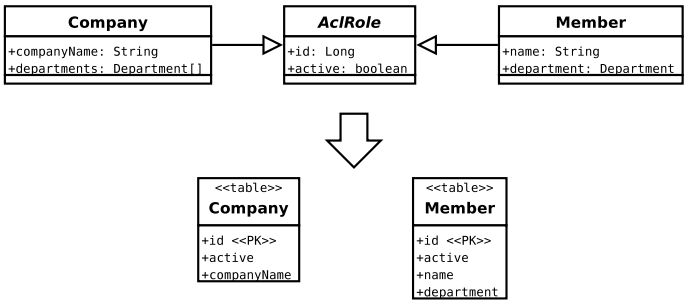
5.4 Implementace perzistenční vrstvy

Pro implementaci perzistenční vrstvy byl zvolen framework Hibernate. Ten přímo podporuje mapování dědičnosti tříd do databáze. Zvolit vhodnou mapovací strategii bylo prvním úkolem. Hibenate podporuje čtyři různé přístupy, jak mapovat hierarchii tříd do databáze:

1. *Table per concrete class with implicit polymorphism* spočívá v mapování každé (neabstraktní) třídy do samostatné tabulky. Každá tabulka obsahuje veškeré atributy dané třídy, tedy včetně zděděných atributů. Toto mapování je výhodné ve své jednoduchosti, přináší ale kompilikace při dotazování na více tříd (je nutné provést několik samostatných dotazů do databáze), polymorfických asociacích nebo při změnách rodičovských tříd, které se musí promítnout do definice všech databázových tabulek.

2. *Table per concrete class with unions* je podobné předchozímu mapování, odstraňuje ale klíčové nevýhody. Třídy jsou opět mapovány do samostatných databázových tabulek obsahujících všechny (i zděděné) atributy třídy. Struktura mapování v konfiguraci Hibernate ale již odpovídá hierarchii tříd, tzn. že „sdílená“ data tříd jsou mapována pouze jednou a pro dotazy se využívá SQL `UNION SELECT` příkazu, který databázové systémy provádějí většinou velmi efektivně. Toto mapování tedy podporuje polymorfické asociace a je vhodné pro implementaci „ploché“ hierarchie tříd.

   Při implementaci systému bylo využito toto mapování pro hierarchii tříd s předkem `AclRole`. Ukázka tohoto mapování je na obrázku 5.4.



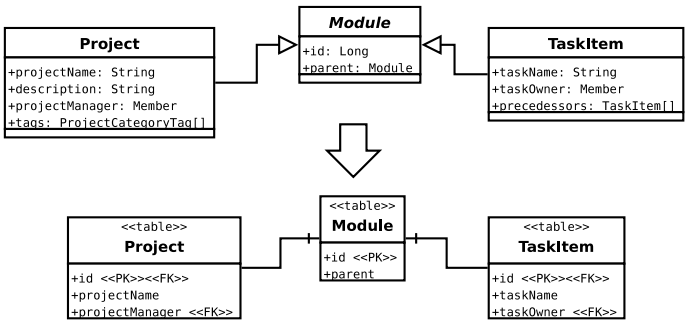
Obrázek 5.4: Hierarchie tříd mapovaná do databázových tabulek metodou *Table per concrete task with unions*

3. *Table per class hierarchy* mapuje všechny třídy v hierarchii do jedné databázové tabulky. Tato tabulka obsahuje všechny atributy ze všech tříd v dané hierarchii. Třída a její atributy jsou ze všech sloupců tabulky vybrány na základě tzv. popisujícho sloupce. Tuto metodu nelze použít, pokud některá z tříd vyžaduje omezení `not-null` u některého z atributů. Další nevýhodou tohoto mapování je větší náročnost na

paměťový prostor, protože každá instance třídy zabere celý jeden řádek v tabulce. Výhodou je naopak největší rychlost při zpracování databázových dotazů. Toto mapování nebylo při implementaci využito.

4. *Table per subclass* reprezentuje hierarchii tříd formou asociací cizích klíčů. Každá třída je mapována do své databázové tabulky a tato tabulka obsahuje pouze atributy definované v dané třídě. Toto mapování nejpřesněji odpovídá objektovému modelu tříd, umožňuje polymorfické kolekce a případné změny v atributech jedné třídy se projevují pouze do jedné databázové tabulky. Nevýhodou tohoto mapování jsou složitější a náročnější databázové dotazy – pokud se nedotazujeme na konkrétní třídu (hledáme instanci abstraktního předka), použije Hibernate SQL příkaz s využitím `OUTER JOIN` přes všechny tabulky v celé hierarchii tříd (!). Pro dotaz na konkrétní třídu `INNER JOIN`[1].

   Tento přístup byl využit při implementaci mapování dědičnosti tříd, které jsou potomky abstraktní třídy `Module`, jak ukazuje obrázek 5.5.



Obrázek 5.5: Hierarchie tříd mapovaná do databázových tabulek metodou *Table per subclass*

Výběr konkrétní strategie pro mapování dědičnosti vždy závisí na zhodnocení všech okolností i dalšího vývoje projektu. Nejuniverzálnější a z pohledu relačního modelu „nejčistější“ mapování *Table per subclass* je bohužel zároveň nejvíce výkonově náročné. Nepřináší žádná omezení jako některá jiná mapování. Volba mapovací strategie je proto vždy kompromisem mezi efektivitou získávání dat z databáze a možnostmi dalšího rozšiřování a úprav. Ukázka konfigurace mapování třídy `Project` pro nástroj Hibernate je v příloze A.

5.5 Implementace zabezpeční

Doporučovaným [6] nástrojem pro zajištění bezpečnosti v aplikacích využívajících Spring Framework je open-source framework *Acegi Security System for Spring*[18]. Podle [6] musí každá internetová aplikace obsahovat bezpečností opatření ve čtyřech oblastech:

autentizace uživatele,

web-request zabezpeční, tzn. zabezpeční různých oblastí internetové aplikace na úrovni HTTP dotazů podle adresy dotazované služby,

[18] `http://www.acegisecurity.org`

service layer security pro autorizování přístupu k jednotlivým službám, které poskytuje aplikační vrstva aplikace,

domain object instance security představuje zabezpeční přístupu a manipulaci s jednotlivými doménovými objekty pomocí ACL (Access Control List).

Acegi Security poskytuje podporu pro všechny čtyři oblasti, a obdobně jako celý Spring Framework tak činní neinvazivním způsobem, to znamená že umožňuje programátorovi vybrat, které oblasti hodlá využít, případně implementovat moduly implementující rozhraní definované Acegi Security a tak zkombinovat vlastní části kódu s Acegi Security.

Při implementaci zabezpečení byla podporu Acegi Security využita pro autentizaci uživatele a web-request zabezpeční. Naopak kontrola přístupu k doménovým objektům a aplikační vrstvě aplikace je přenesena do *business interface* rozhraní. Důvodem byla nedostatečná podpora Acegi Security pro přístupová práva k doménovým objektům v implementované aplikaci (uživatelské role se mění u každého projektu – jeden uživatel aplikace může mít v projektu A práva pouze ke čtení, v druhém projektu ale může mít roli projektového manažera a plná práva pro správu celého projektu). Autorizaci uživatele pro přístup k jednotlivým doménovým objektům kontroluje třída `AclManager` popsaná v kapitole 4.3.3.

Využití Acegi Security přináší jednu podstatnou výhodu pro nasazení projektu. Abstrakce, kterou poskytuje Acegi Security umožňuje snadno zaměnit autentizační část za zdroje jako jsou adresářové služby LDAP, single-login nástroje (např. open-source CAS[19]), podporu pro X509 certifikáty a další.

5.6 Rozsah implementace

V rámci diplomové práce jsem implementoval následující části navrženého informačního systému:

- jádro systému zajišťující kontrolu přístupových práv, uživatelských účtů a přístupu k jednotlivým částem systému,
- podporu pro více oddělených prostorů v rámci jedné instalace aplikace (SaaS),
- správu organizační struktury – rozdělení na Company a Department,
- vytváření a modifikaci projektů,
- nástroj Label pro kategorizaci projektků a následnou podporu vyhledávání podle definovaných nálepek,
- nástroj Task, včetně optimalizace projektu metodou kritické cesty, výpočtu časových rezerv a časové rozložení zdrojů,
- nástroj Message včetně podpory komentářů.

[19]CAS – Central Authentication Service, `http://www.ja-sig.org/products/cas`

Kapitola 6

Rozšíření systému

Výběr Java platformy měl pozitivní vliv na množství dostupných technologií, které umožní další rozšíření implementovaného systému. Nejzajímavější oblast dalšího vývoje implementováneho systému popisují následující odstavce.

6.1 Rozšířená podpora sdílení souborů

Content Repository for Java Technology API (JCR) je specifikace standardního aplikačního rozhraní pro přístup k systémům pro správu dokumentů. Tento standard byl přijat v roce 2005 jako JSR 170[1] s cílem sjednotit přístup k různým implementacím systémů pro správu dokumentů.

API definuje základní požadavky na vlastnosti datového uložiště a metody pro přístup k datům uloženým v informačním systému pro správu dokumentů. Data jsou uložena v systému v tzv. uzlech, které jsou uspořádány do hierarchické struktury a každý uzel má sadu vlastností. Uložiště může být rozděleno do více nezávislých struktur – *workspace* a podpora jmenných prostorů zabraňuje kolizím jmen a umožňuje odkazovat na uzly v jiných workspace. Standard definuje podpopru pro vyhledávání dat, verzování dokumentů, zajištění přístupových práv, dotazy pomocí XPath syntaxe nebo transakce.

Integrace implementovaného systému s CMS systémem přinese větší možnosti pro využití nástroje *File* (viz. analýza požadavků v kapitole 3.1.8). Největšími přínosy bude možnost fulltextového indexovaného vyhedávání v uložených datech (včetně souborů ve formátu Portable Document Format, Microsoft Word, Excel, PowerPoint, OpenDocument využívaný OpenOffice.org a další), a podpora pro verzování dokumentů.

V současné době existují open-source i komerční implementace JCR. Nejrozšířenější open-source implementací je Apache Jackrabbit[2]. Spring Modules[3], podprojekt Spring Frameworku, obsahuje modul pro integraci Apache Jackrabbit do aplikací využívajících Spring Framework. Způsob integrace velmi podobný např. integraci Hibernate nebo JDBC datového zdroje v aplikačním kontextu aplikace.

Apache Jackrabbit vyžaduje pro spuštění JEE aplikační server nebo Apache Tomcat. Jako perzistenční vrstvu lze využít databázový server. Integrace tohoto systému tak nepřináší, kromě výkonových, žádné další požadavky na softwarové vybavení serveru.

[1] JSR – Java Specification Request, `http://jcp.org/en/jsr/detail?id=170`
[2] `http://jackrabbit.apache.org`
[3] `https://springmodules.dev.java.net`

Kapitola 7

Zhodnocení práce a závěr

Tato diplomová práce zpracovává projekt z oblasti projektového řízení. Protože se jedná o velmi rozsáhlou oblast, věnuje se první kapitola nejen představení samotného projektového řízení a navazující oblasti týmové spolupráce, ale představuje také organizační strukturu a projekty neziskové studentské organizace IAESTE České republiky, která slouží jako vzorový příklad organizace, ve které je vhodné implementovat aplikaci podporující projektové řízení a týmovou spolupráci.

Po specifikaci požadavků v druhé kapitole a důkladné analýze v kapitole třetí, zpracovává čtvrtá kapitola návrh systému, který podpoří všechny klíčové oblasti projektového řízení s důrazem na další rozšiřování systému. Tato část diplomové práce byla spolu s výběrem vhodných technologií stěžejní. Pro vývoj aplikace byla vybrána platforma Java Enterprise Edition, která spolu s podpůrnými open-source nástroji (Spring Framework, Hibernate, Acegi Security) tvoří základ, na kterém lze stavět robustní a snadno rozšiřitelné aplikace.

Součástí diplomové práce je implementace klíčových oblastí navržené aplikace – jedná se o oblast definování organizační struktury aplikace a uživatelských účtů, zajištění bezpečnostních pravidel, vytváření a optimalizaci projektů a komunikaci mezi členy projektového týmu.

Implementovaný systém je navržen pro další rozšiřování. Nejbližší vývoj by se měl zaměřit na implementaci dalších nástrojů navržených v kapitole 3.1 a dále, na základě zkušeností uživatelů s ovládáním systému, optimalizovat uživatelské rozhraní aplikace.

Hlavním přínosem práce je představení nového vztahu mezi nástroji podporující projektové řízení a nástroji podporujícími týmovou spolupráci. Tyto dvě oblasti se podařilo integrovat do jednoho celku, který uživatelům nevnucuje vlastní filozofii práce na projektech, ale poskytuje vzájemně provázané nástroje, které usnadní a zpřehlední každodenní práci.

Literatura

[1] Christian Bauer. *Java Persistence with Hibernate*. Manning Publications Co., 2006. ISBN: 1-932394-88-5.

[2] Tom Brick. Usability first: Groupware. `http://www.usabilityfirst.com/groupware` (květen 2007).

[3] Petr Glos Filip Procházka. Přístupová práva (nejen) v IS BAPS a jak na ně. *Zpravodaj ÚVT MU*, XIV(3):11–14, 2004. ISSN 1212-0901.

[4] Bent Hansen. *Project Management for Interntional Students, rev. 06*. Odense University College of Engineering, 2006.

[5] Chris Hendrickson. Project management for construction, second edition. `http://www.ce.cmu.edu/pmbook/` (květen 2007), 2000.

[6] Rod Johnson. *Professional Java Development with the Spring Framework*. John Wiley & Sons, 2005. ISBN: 978-0-7645-7483-2.

[7] Milton D. Rosenau Jr. *Successful Project Management, Third Edition*. John Wiley & Sons, Inc., 1998. ISBN 0-471-29304-0.

[8] Milton D. Rosenau Jr. *Řízení projektů*. Computer Press, 2003. ISBN 80-7226-218-1.

[9] Robert V. Stumpf. *Object-Oriented System Analysis and Design with UML*. Prentice Hall, 2004. ISBN 0131434063.

Dodatek A

Objektově relační mapování třídy Project

Následující kód ukazuje, jakým způsobem je konfigurováno objektově relační mapování třídy `Project` do databáze prostřednictvím frameworku Hibernate.

```
<?xml version='1.0'?>
<!DOCTYPE hibernate-mapping PUBLIC
    '-//Hibernate/Hibernate Mapping DTD 3.0//EN'
    'http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd'>
<hibernate-mapping package='com.blbeckove.viitan.hammam.base'>

    <joined-subclass name='Project' table='PROJECT' extends='Module'>

        <key column='ID'></key>

        <property name='name' column='PROJECT_NAME'></property>
        <property name='description' column='PROJECT_DESCRIPTION'></property>
        <many-to-one name='department' class='Department'
            column='DEPARTMENT_ID' not-null='true'>
        </many-to-one>
        <many-to-one name='projectState' class='ProjectState'
            column='PROJECTSTATE_ID' not-null='true'>
        </many-to-one>
        <one-to-one name='projectGroup' class='ProjectGroup'></one-to-one>
        <many-to-one name='projectManager' class='Member'
            column='PROJECT_MANAGER_ID' not-null='false'>
        </many-to-one>
        <property name='startDate' type='timestamp'
            column='START_DATE'>
        </property>
        <property name='closeDate' type='timestamp'
            column='CLOSE_DATE'>
        </property>

        <set name='projectMembers' table='MEMBER_PROJECT'>
            <key column='PROJECT_ID'></key>
            <many-to-many column='MEMBER_ID' class='Member'></many-to-many>
        </set>

        <one-to-one name='taskModule' class='TaskModule'
            property-ref='project'>
        </one-to-one>

        <one-to-one name='messageModule' class='MessageModule'
            property-ref='project'>
        </one-to-one>

        <set name='tags' table='PROJECT_PROJECTCATEGORYTAG'>
```

```
39              <key column='PROJECT_ID'></key>
40              <many-to-many class='ProjectCategoryTag' column='CATEGORYTAG_ID'/>
41          </set>

42      </joined-subclass>
43  </hibernate-mapping>
```

Dodatek B

Adresářová struktura projektu

`/src` – zdrojové soubory aplikační logiky informačního systému

- `com.blbeckove.viitan.hammam.base` – doménové (POJO) třídy informačního systému
- `com.blbeckove.viitan.hammam.business` – rozhraní třídí zajišťujících aplikační logiku
- `com.blbeckove.viitan.hammam.business.hibernate` – implementace rozhraní s využitím Hibernate
- `com.blbeckove.viitan.hammam.controller` – implementace MVC Controlleru
- `com.blbeckove.viitan.hammam.controller.command` – *command* třídy pro využití v rámci formulářů Spring MVC
- `com.blbeckove.viitan.hammam.controller.validator` – třídy pro validaci *command* objektů při zpracování formulářů
- `com.blbeckove.viitan.hammam.dao` – rozhraní pro perzistenci doménových objektů
- `com.blbeckove.viitan.hammam.dao.hibernate` – implmentace rozhraní s využitím Hibernate
- `com.blbeckove.viitan.tags` – implementace vlastních JSP tagů (viz. 5.2)

`/WebContent` – zdrojové soubory internetové aplikace

- `css` – definice kaskádových stylů
- `img` – obrázky a ikony
- `WEB-INF/jsp` – sobory JSP stránek
- `WEB-INF/lib` – definice vlastních JPS tagů
- `WEB-INF/applicationContext-acegi-security.xml` – konfigurační soubor Acegi Security for Spring
- `WEB-INF/applicationContext.xml` – konfigurační soubor Spring Frameworku

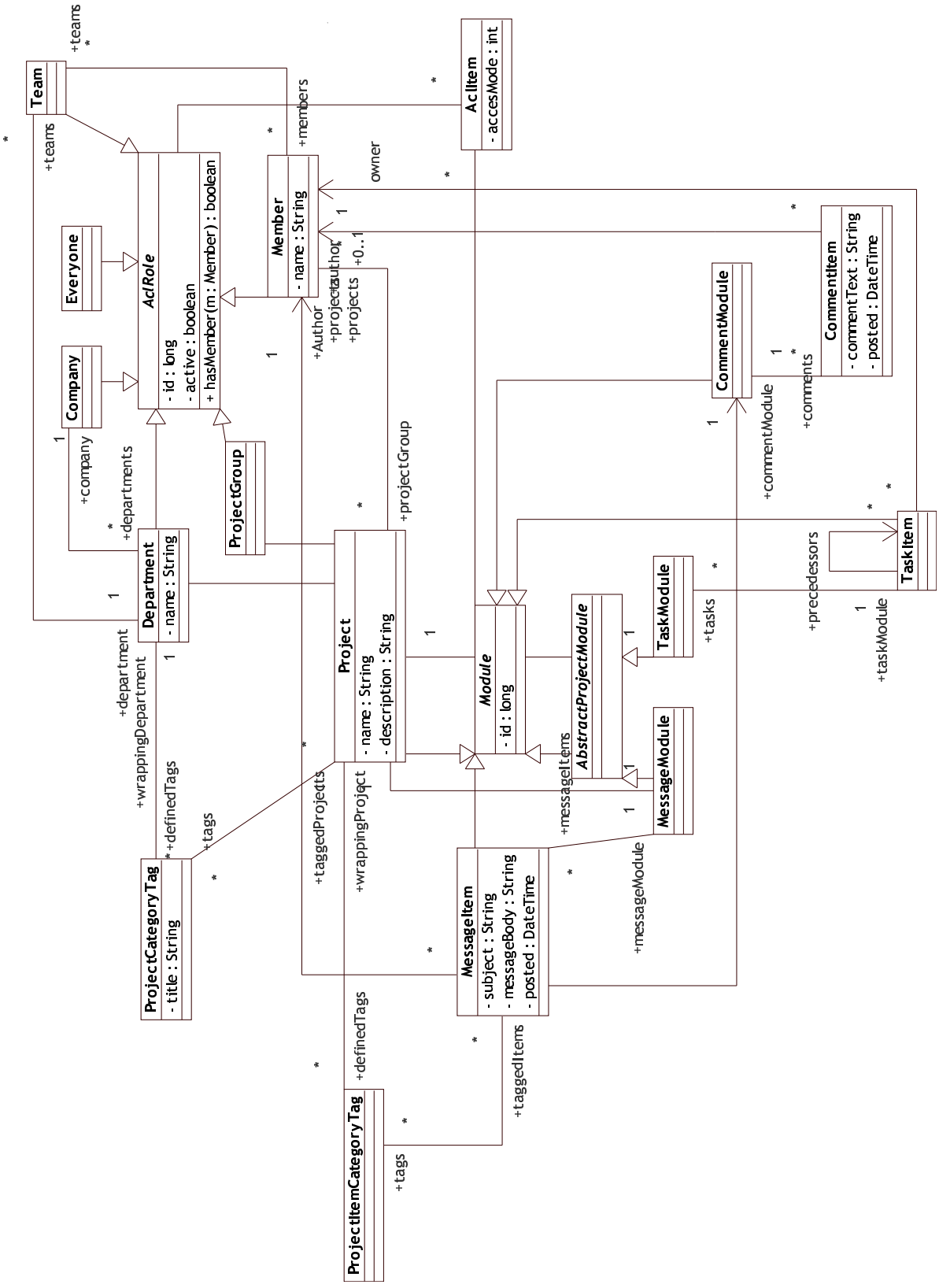
– `WEB-INF/hammam-servlet.xml` – konfigurace DispatcherServrletu, který zajišťuje obsluhu všech příchozích HTTP požadavků

– `WEB-INF/web.xml` – konfigurační soubor webové aplikace (podle JEE standardu).

Dodatek C

Class diagram

Obrázek C.1 zobrazuje diagram tříd implementovaného systému.



Obrázek C.1: Class diagram implementované aplikace